IDACT TRANSFORMATION MANAGER

by

Brian Hay

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

Doctor of Philosophy

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

November 2005

APPROVAL

of a dissertation submitted by

Brian Hay

This dissertation has been read by each member of the dissertation committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

Michael Oudshoorn

Approved for the Department of Computer Science

Michael Oudshoorn

Approved for the College of Graduate Studies

Joseph J. Fedock

STATEMENT OF PERMISSION TO USE

In presenting this dissertation in partial fulfillment of the requirements for a doctoral degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library. I further agree that copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for extensive copying or reproduction of this dissertation should be referred to ProQuest Information and Learning, 300 North Zeeb Road, Ann Arbor, Michigan 48106, to whom I have granted "the exclusive right to reproduce and distribute my dissertation in and from microform along with the non-exclusive right to reproduce and distribute my abstract in any format in whole or in part."

Brian Hay

November 2005

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (CONTINUED)

ABSTRACT

As scientific models and analysis tools become increasingly complex, they allow researchers to manipulate larger, richer, and more finely-grained datasets, often gathered from diverse sources. These complex models provide scientists with the opportunity to investigate phenomena in much greater depth, but this additional power is not without cost. Often this cost is expressed in the time required on the part of the researcher to find, gather, and transform the data necessary to satisfy the appetites of their data-hungry computation tools, and this is time that could clearly be better spent analyzing results. In many cases, even determining if a data source holds any relevant data requires a time-consuming manual search, followed by a conversion process in order to view the data. It is also commonly the case that as researchers spend time searching for and reformatting data, the expensive data processing hardware remains underutilized. Clearly this is not an efficient use of either research time or hardware. This research effort addresses this problem by providing a Transformation Manager which leverages the knowledge of a group of users, ensuring that once a method for a data transformation has been defined, either through automated processes or by any single member of the group, it can be automatically applied to similar future datasets by all members of the group. In addition to the use of existing transformation tools for well known data formats, methods by which new transformation can be automatically generated for previously unencountered data formats are developed and discussed. A major deliverable for this work is a prototype Transformation Manger, which implements these methods to allow data consumers to focus on their primary research mission, as opposed to the mechanics of dataset identification, acquisition, and transformation.

1

PROBLEM STATEMENT

Overview

> "The heterogeneous, distributive and voluminous nature of many government and corporate data sources impose severe constraints on meeting the diverse requirements of users who analyze the data." [HaNa00]

The ever-increasing rate at which data is being collected and stored greatly overwhelms the current ability to make these valuable data resources easily available to a wide range of data consumers. At one extreme, these consumers include scientific researchers, whose increasingly complex models manipulate larger, richer, and more finely-grained datasets gathered from multiple heterogeneous sources, providing the opportunity to investigate phenomena in much greater depth than has ever been previously possible. At the other end of the spectrum are decision-makers, policy-makers and members of the general public, who may be concerned with data more closely related to their personal interests, such as the level of contaminants in animal species in their region, or studies describing air quality. Despite different levels of technical knowledge, this vast continuum of data consumers faces similar problems, namely how to locate, assemble, and integrate heterogeneous domain-specific data into a format that meets their specific needs. For the technically savvy data consumer, this task is possible today, but often only with significant and time-consuming manual data manipulation efforts, using time that could be better spent in the data analysis process. However, the typical data consumer may possess only some, if any, of the knowledge necessary to locate relevant data, let alone convert and integrate it into a useful product. The ability to view products

based on multiple heterogeneous datasets in a new and novel manner is often the key to enhancing the global knowledge base, and increasing scientific understanding. However, this opportunity is frequently unavailable to data consumers, either due to time constraints, or insufficient access to data resources.

The current situation demonstrates an ineffective use of our vast and ever-expanding data resources, and of the time and efforts of data consumers. As a result, while data that could be used to answer important questions often exists, it remains beyond the reach of most data consumers, and the scientific issues that that the data could help address remain partially or fully unresolved. Furthermore, data collected at great cost for one project or purpose often becomes less useful over time, not due to data quality issues or a lack of interest in the topic, but simply because other data consumers often do not know it exists, and could not access it in a useful manner if they did. A solution must be found to automate much of this process, allowing for increased productivity on the part of scientists, greater data access for all levels of data consumers, and increased throughput and utilization of models and analysis tools.

Detailed Problem Description

While science has traditionally been driven in large part by data analysis, the increasingly widespread use of electronic resources such as personal computers, database management systems, and advanced modeling tools allows analysis at a more detailed level than has ever been possible in the past. Similar technological advances are also being applied to the decision making process in many other fields. For example,

businesses often gather and analyze data to determine how to more effectively market products or provide better service to their customers, and political entities are increasingly dependant on the analysis of polling data.

These approaches can only be effective, however, if the data which the analysis tools ingest has been identified, collected, organized, and made available. Data collection is typically the least of these problems, with data being collected in massive quantities on almost every aspect of human life and the environment in which we live. From consumer preference in grocery stores and television viewing habits, to satellite remote sensing and in-situ sensors, there is a staggering amount of data being collected on a daily basis. This wealth of data will remain largely untapped if it cannot be analyzed and transformed into knowledge, and it is often a failure in these areas that results in data not being used as effectively as it could be.

The disconnect between data archivists, who are responsible for storing datasets, and data consumers, such as scientists, researchers, or policy-makers, is often the result of several factors, including unorganized data, data format issues, incomplete or missing metadata, or data archives whose existence is not widely known. The first three of these issues are directly relevant to this work, while the fourth issue is addressed in part by IDACT, of which this work is a component.

Unorganized Data

The first issue which affects data consumers is that the data is often not organized, or is stored in manner which is not conducive to searching. Data is frequently collected

for a specific purpose, and once it has been used in that limited fashion the raw data is archived without thought to whether it may be useful beyond this initial application. These "data piles" often contain valuable data that is typically underutilized outside the original project which motivated its collection. Often the expense or effort required on the part of a data consumer to determine if the archive contains data germane to their research, and to then extract it is prohibitive. As a result, data collection efforts may be duplicated, or historical datasets may not be considered in comparison with more recent or more detailed measurements due to the associated costs or time requirements.

Data Format Issues

Even in cases where data is organized, it is frequently the case that the data is stored in a format which is not useful to the data consumer. There have been many attempts to address this issue through the use of standard formats, [Cov01, Cov05, NWS05, ThSh04, W3C05a, W3C05b, W3C05c], but in many cases these have met with limited success. "Standard" formats tend to fall into one of two categories, both of which have significant problems. The standards which allow for flexibility [NCS05a, Uni05, ITS05] are often also very complex and unwieldy, and as such have a steep learning curve that is unattractive or prohibitive for many data consumers and owners. The other extreme includes the narrowly defined standards, which are frequently so inflexible that an attempt to collect even a single new field results in the standard being re-written, resulting in multiple versions of the "standard", which clearly defeats the objective [ICE05]. While some useful domain specific standards do exist, they still do not address

the issue of data consumers whose data fusion tools draw datasets from multiple domains, which is an increasingly common and useful analysis approach across many research domains [Ada94, Bay03, Bro02, CRS91, Das97, FoBo98, Hall97, KJW02, Rus02]

There are also many cases where there are several, competing, and applicable "standards" from which a data owner could select a storage format. For example, the Federal Geographic Data Committee which supports at least seven geospatial data standards [FGD05] appears to be distinct from the Environmental Data Standards Council [EDS05] although the data needs of the two seem compatible. Storing multiple copies of the data, with one copy in each format, is not an advisable solution. Such an approach is not only prohibitive in terms of storage space, but also introduces the problem of data integrity, where updates are made in one copy, and not propagated to the others. Even if the data archivist could select the "best" format (which is unlikely to exist), there is no guarantee that a better format will not be available tomorrow. For data which has value in the future (such as benchmark scientific data which can be used to show trends over time), making the data accessible today is insufficient; it must remain available over its useful lifetime, despite any changes in data archiving formats which cannot be predicted at this time. As part of the SynCon project [HaNa00, Nan97a, Nan97b, Nan97c, Nan97d, NaWi98, NWG99a, NWG99b, NaSt00, RaNa98, TeNa98, Rei01], scientists were queried about their requirements for a data center, and one of the most common desires was make their data available in whatever format it is needed forever [NaHa01]. As such, it is likely that no single data format will meet the needs of all data consumers.

A related issue that often plagues data centers is the data submission process. One common goal of a data center is to collect as much relevant data as possible, and as such they must determine how data collectors, such as scientists, must submit their datasets. Many current implementations involve the definition of an input format, which is either similar to, or can be easily converted to, the logical storage representation employed by the data center. In practice this is rarely an effective approach, because data collectors generally store data in a format that meets their immediate needs, rather than in a format that meets the submission requirements of the data center. Often when confronted with the issue of submitting data to a data center in a required format, data collectors will choose not to submit datasets unless compelled to by some external force, such as the submission being a requirement for continued funding. During the development of the SynCon system, several international data centers facing just this problem were encountered, with the results being that data from the European countries, where funding was dependant on data submissions, was plentiful, and data from other nations was scarce at best, despite there having been many valuable data collection efforts underway. The scientists polled during the SynCon requirements phase also cited "allowing data collectors to submit data in any format they chose" [NaHa01] as being a significant issue in their efforts to submit data to data centers.

While tools do exist that perform automated conversions between some common formats in specific domains [NCS05b, Dat05, ICO05], these typically require the data consumer to have a high level of technical proficiency, in that they must identify, acquire, install, and run the appropriate tool for a given conversion task. Unfortunately, these

tools generally do not provide any assistance when either the source or desired data formats are less common. Even between commonly used data formats there may not be a suitable conversion tool available. Furthermore, if such a conversion tool exists, it must often be used merely to determine if a given dataset is relevant to the issue being explored by the data consumer. While some researchers are proficient in the use of computing resources, and could even write their own programs to convert data as necessary, any time that they spend in this effort detracts from their primary mission of scientific data analysis. For those whose level of technical expertise is not sufficient to use or write conversion tools, they can either fund computer experts to perform those tasks, or limit their analysis to datasets that are more accessible, but possibly less useful. In order for the full potential of a given dataset to be realized, it is vital that it be accessible to users who do not have an in-depth knowledge of computing resources such as operating system administration, programming languages, and database systems.

Incomplete or Missing Metadata

The third issue commonly encountered by data consumers is that access to a dataset is usually insufficient if there is no access to the associated metadata. There are an increasing number of formats in which metadata can be embedded in or reliably associated with the dataset. Examples of such formats include NetCDF [Uni05], or many XML-based formats [Cov01, Cov05, NWS05, ThSh04, W3C05a, W3C05b, W3C05c]. However, in many cases the complete metadata for a dataset is not available, which reduces the usefulness of the data, or even renders it useless. An example of such a case

involves the Outer Continental Shelf Environmental Assessment Program (OCSEAP) program, in which large quantities of environmental data was collected in the Bering Sea, the Beaufort Sea, and the Gulf of Alaska in the 1970s [US91]. The primary method of data access for the data was via a custom written software package, and the majority of the data itself was stored in a custom format. However, the software package was poorly documented, is difficult to locate, and does not work on current operating systems. As a result, while much of the data remains in the custom format, there is no indication of what it means or how to extract it, so the bulk of this important dataset, which could provide an excellent historical baseline for future studies, has been lost.

During the SynCon system development, several datasets were encountered in which the data collectors stored the data collectors in spreadsheet documents (typically Microsoft Excel), and for which the metadata only existed in the mind of the originating scientist, or as a heading of a spreadsheet column. While this method was initially acceptable while the data was being used by the data collector, it essentially made the dataset useless to any other data consumer. In such cases, metadata can sometimes be recovered through discussion with the data collector (if available), or by analysis of associated documentation, such as journal articles. However, such inefficient approaches are typically expensive in terms of time and/or money.

Problem Statement

The problem that is addressed in this research effort is that of providing automated mechanisms for transforming datasets between data formats, thereby reducing

the burden, in both time and technological skill, currently placed on data consumers as they attempt to access existing datasets. In addition, the related issue of allowing data collectors to more easily submit datasets to a data center is also addressed. The result of this is to allow data consumers and collectors to focus on their areas of expertise, rather than the technical details of dataset access, while increasing the range of datasets, in their problem domain, available for their use.

While this effort initially targets scientists at the University of Alaska Fairbanks (UAF) and NASA, the conceptual approach developed in this work, and the open source Transformation Manager which implements that approach, will benefit to a far broader spectrum of users from the scientific, governmental, and commercial domains by allowing data producers and consumers to focus on the use of data rather than the details of data management and manipulation.

FOUNDATIONAL WORK AND CURRENT APPROACHES

General Foundational Work

This work builds on general foundational work in the areas of middleware and data portals, integration of heterogeneous schemas, transformation tools and transformation components, and ongoing research efforts in data organization.

Middleware and Data Portals

This work addresses, in part, some of the current issues related to *data portals*, which promise access to vast underlying data holdings via a single seamless user interface, such as a web page operating in conjunction with a middleware layer [CEH04, DMR03, WCH02, WSW05]. However, this promise is far from being fully realized. Recent research indicates that data portals are currently not effective at dynamically combining data sources from heterogeneous source. For example, in some cases primitive hardwired approach that is specific to a very small set of carefully chosen data types/sources [GNS00], while other approaches involve the use of multiple "portlets" on a single web page [PrGu03], each of which queries only a single data source in isolation.

Integration of Heterogeneous Schemas

Several recent works address the issue of the integration of heterogeneous schemas, as it has become an important field of study with applications in many domains, such as web-service interactions, data discovery, management, processing and fusion.

The most basic version of this process is limited to schemas for which there is at most a one-to-one association between fields [CAD01, DMD03, EJX01, LiCl00, MBR01, MMR02, PVH02, RaBe01]. For example, the SEMINT system [LiCl00] uses a neural network to determine which pairs of fields in two database schemas are equivalent. However, such one-to-one approaches ignore the issue of more complex relationships between fields, in which, for example, a set of fields from one schema may be equivalent to another set in a second schema. As a result, such simple systems are rarely valuable for use with real-world schemas, which typically feature more complex mappings.

Some researchers have addressed the issue for schemas which feature more complex associations [DLD04, DMD03, LiCl00, MBR01, MMR02, PVH02, RaBe01]. For example, Embley et al [EXD04] proposed an approach which in large part relies on the predefinition of static list of synonyms for a given domain, which is only useful if such a list can actually be generated. In many cases, more flexibility is required to ensure that the system adapts to changes in the domain. The CUPID system [MBR01] also addresses the definition of more complex associations between heterogeneous schemas, but does not provide any mechanism by which the nature of the association is defined. For example, while CUPID may identify an association between the *Line* field in one schema and the *ItemNumber* field in aonther, it does not address the issue of how values from the *Line* field can be transformed into equivalent values in the *ItemNumber* field. In addition, none of these systems are capable of generating a new transformation method once associations have been defined, which is a major focus of this research effort.

Transformation Tools

Some transformation tools for individual data source and target formats have already been defined in many domains [NCS05b, Dat05, ICO05], and are often available via web or FTP sites. However, these tools typically involve standard data formats for the source and target, so while they are useful in many cases, they are by no means a universal solution. In addition, it is often the responsibility of each individual data consumer to locate, acquire, install, and operate these tools, rather than leveraging the setup work performed by one user to benefit all data consumers in a given domain.

Other transformation tools take the form of custom programs or scripts, written in-house at data repositories, processing centers, and distribution sites, or by individual data producers, archivists, or consumers. These tools generally fit a given need for which no generally available tool can be located, but the work required to produce them is often repeated by others facing similar problems, resulting in a duplication of effort.

Transformation Components

The XSLT Standard Library [Sou05] provides a large number of XSLT components, which can be utilized in the construction of complete XSLT documents. These foundational components can be incorporated into the custom transformation creation process discussed in chapter 5.

Foundational Work

This effort primarily builds on the work performed as part of the SynCon Project at the

University of Alaska Fairbanks. In particular, the SIMON agent created as part of that program forms a basis for the intelligent transformation component of this work [Hay00, HaNa00].

SynCon Project Data System

The Arctic Monitoring and Assessment Programme (AMAP) is an international organization comprised primarily of representatives from government and the scientific communities in the eight circumpolar arctic countries. The objective of AMAP is to:

> "[provide] reliable and sufficient information on the status of, and threats to, the Arctic environment, and [provide] scientific advice on actions to be taken in order to support Arctic governments in their efforts to take remedial and preventive actions relating to contaminants" [AMA05].

At the 1997 AMAP meeting, participants from around the world were dismayed by the absence of U.S. environmental data. When the 1998 AMAP Assessment Report: Arctic Pollution Issues [AMA98] was published this issue was still a matter of grave concern for the circumpolar arctic scientific community, and an embarrassment to U.S. scientists and funding agencies. The primary reason for this omission was that the AMAP submission process and required formats were too complex and time-consuming for the U.S. scientists, who typically have neither the time nor the inclination to perform the tedious conversion process from their raw data formats.

One example of this problem is demonstrated by the data submission process at the AMAP Marine Thematic Data Center operated by the International Council for the Exploration of the Sea in Copenhagen, Denmark [ICE05a]. The submission process

requires the data producer to follow an extremely strict data format, which has evolved over time as the requirements associated with the datasets being collected have changed. The storage format uses input fields based on fixed format lengths, represented in large part by the reference codes that are used in the underlying data storage format. For example, there are 16 codes, such as "CLAV ELL" and "PSEU TUM" which can be used to indicate the presence of a given fish disease, which must be included in a field title "DISEA" [ICE05b]. Failure to follow this requirement in even a single record results in the rejection of the entire dataset, and there are many other fields with similarly cryptic codes.

In response to this issue, the SynCon Project staff at the University of Alaska Fairbanks pledged to work with AMAP and the U.S. scientists to help resolve this issue. In an initial step, a group of international scientists involved in environmental research related to the Arctic environment were asked what was needed to encourage them to submit their data to a central facility. Their responses indicated that as a group they had the following basic requirements [NaHa01]:

- Accept my data in its current format – each scientist typically had a data storage format that they were comfortable with (many of which involved spreadsheet documents), and they wanted to submit datasets to data centers in those formats.
- Don't make me do much extra work – none of the scientists were interested in data submission processes which required significant additional effort on their part, as it detracted from their primary scientific mission.

- Don't make me hire a data specialist – the scientists typically did not have funding to hire additional staff to perform data manipulation operations, and diverting any funding they did have was viewed as reducing the funding for their scientific operations.
- Don't ask me too many questions – there was a desire that any operations on the part of the data center to ingest data submissions should happen with as little interaction as possible between the data center and the scientists.
- Make my data available in the formats I want and will want everyday, forever – typically the scientists were aware that their needs for data formats may change in the future, and they wanted to ensure that a data center could provide them with data in the formats which they required not only today, but also in the future as data formats evolved.
- Protect my data from unauthorized access – while the scientists were generally very interested in having access to the data collected by others, they were very wary about others having access to their data, at least without their permission.

The SynCon development team successfully met this challenge for the AMAP community, in large part through the development of automated processes and an intelligent agent [Hay00, HaNa00] that is able to accept diverse data formats as input and translate the data into the underlying storage format of the data system. In addition, the data consumer can access data via a variety of interfaces which provide abstraction from the raw storage format, ensuring that the output formats can change to meet the needs of

the user, without impacting the underlying data system. Data collectors were also able to specify restrictions on the distribution of the data, including the date at which it would be publicly available, and at what level of detail it would be released. For example, a data collector could specify that *catalog* data, which describes the dataset in broad terms, be made publicly available on a given date, while restricting the release of the individual measurements, known as the *detail* data, until a later date. A third classification that was frequently utilized by the AMAP community was *aggregate* data, with an associated release date which indicated when data products based on the detail data could be published.

Due to these developments, scientists could concentrate on their primary mission, and the data manipulation process, which would have required hours, days, or even weeks of effort for the scientists to complete for other AMAP data systems, was accomplished extremely efficiently through large-scale automation. As a result, the SynCon Project was selected to serve as two of the five AMAP Thematic Data Centres (TDC). The other three TDCs are based in Europe, and the addition of these two TDCs in North America was seen by many in the AMAP community as a commitment to address the lack of U.S. data in the AMAP data centers.

It has since become apparent that scientists from around the world are choosing to contribute their data to SynCon even when other data centers were their original targets. In a 2001 letter commending the SynCon Project, the AMAP Executive Secretary Lars-Otto Reierson observed that

> [T]he SynCon solution has proven so successful that part of the scientific community that are responsible for reporting data to the AMAP marine

> and atmospheric TDCs [Thematic Data Centers] have preferred to send their data to UAF for incorporation in the SynCon Database instead. This has particularly been the case with the North American scientific community that has no previous experience of reporting to the European-based marine and atmospheric TDCs. UAF is therefore now exceeding it original commitment to AMAP by processing all datasets received. [Rei01]

SynCon is based at the University of Alaska Fairbanks. Although its initial mission was the collection and distribution of geo-referenced environmental contamination datasets, it was designed to be modified to hold any measurement data, and has been proven as an easily extendable system as data needs expand. In particular, the SynCon framework has been applied to serve as the United State Environmental Protection Agency's (EPA) Alaska Heavy Metals Data System.

As a result of the success of SynCon, and the intelligent data processing agents in particular, two members of this research team (Hay and Nance) traveled to Copenhagen, Denmark in 2004 to advise staff at the International Council for the Exploration of the Sea (ICES) [ICE05], which houses the AMAP Marine TDC, on approaches to data transformation and storage for their evolving data center. Of the five AMAP TDCs, ICES has traditionally had the most challenging data submission process, which relies in large part on a very rigid data format. Although many European scientists do currently contribute to data to ICES, it is generally only because it is a requirement of the various funding and government agencies, and typically requires a significant effort on the part of each scientist. As a result of this daunting effort, and the fact the funding agencies in other areas do not impose such strict reporting standards, ICES generally receives very

few submissions from outside Europe, despite there being large quantities of relevant data from other regions.

SIMON

SIMON is an intelligent agent designed as part of the SynCon project [HaNa00]. The primary goal of SIMON is to allow data submissions to be accepted for inclusion in the database with minimum effort on the part of the either the data archivist or the data collector. In particular, SIMON addresses the deficiencies, as cited by data collectors and observed directly observed during the SynCon requirement phase, in the data submission process used by other data centers. These deficiencies include the following two major issues:

- The use of a rigid, data format for all data submissions.
- The rejection of an entire submission if problems were encountered at the field or record level of the dataset.

As a result of the problems encountered during the data submission process, many scientists chose not to contribute data unless compelled to by some external influence, such as it being a requirement of their funding agency or government.

In response, the goals for the development of SIMON were to provide a mechanism which would be flexible in the input formats it accepted, and which would attempt to automatically determine how to process each data submission. The completed SIMON agent was quite successful in meeting these goals, and while it was not capable of automatically fully ingesting all possible data submissions, it performed the majority

of the work for most datasets, despite each submission being in a unique format. To date, SIMON has been used by SynCon personnel to process over 60 datasets, consisting of over 130,000 measurement records. In all of these cases, no specific input format was required, and as a result scientists submitted data in whatever format they found most suitable.

During the conversion process, user guidance was requested if SIMON was not able to determine how to proceed. The user was presented with a dialog box which described the problem, and provided the most likely solutions, at which point the data archivist was free to select the most appropriate approach. SIMON then not only used the user direction to process the current dataset, but also augmented a knowledgebase which allowed the automated processing to be more effective when faced with similar situations in the future.

Not only was SIMON capable of processing heterogeneous input formats, but when problems were encountered in a dataset, such as a lack of data in a required field, it rejected the data at the lowest level possible. This often meant that datasets which had been rejected in their entirety by other data centers were processed by SIMON and accepted in the SynCon data center with only a small number of records returned to the data collector for clarification and resubmission.

The output produced by SIMON includes a report which is returned to the data collector. This report, among other things, provides a method for cyclic refinement of the dataset by listing additional fields which the data collector may want to submit. If such a resubmission is made, the new data can be inserted into the database, and the dataset is

then more useful to data consumers. This encouraged ongoing expansion of the datasets, including the addition of data as new fields were identified or requested by other users.

SIMON is still in use by SynCon, and continues to allow scientists to submit data in a format that meets their needs, rather than the needs of the data center. As a result, SynCon has been successful in archiving and distributing important datasets from scientists who had previously been unable or unwilling to submit data to other data centers.

Other Systems

Atmospheric Radiation Monitoring Program

The Atmospheric Radiation Monitoring (ARM) Program is a U.S. government effort to address global warming research, with a particular focus on

> the crucial role of clouds and their influence on radiative feedback processes in the atmosphere. The primary goal of the ARM Program is to improve the treatment of cloud and radiation physics in global climate models in order to improve the climate simulation capabilities of these models. ARM's scientists research a broad range of issues that span remote sensing, physical process investigation and modeling on all scales. ARM's site operators focus on obtaining continuous field measurements and providing data products to promote the advancement of climate models. [ARM05]

While the ARM system does provide a mechanism for obtaining data in a variety of formats, which attempt to address the needs of data consumers, including the provision of pre-computed data products, there is no mechanism by which a user can easily define a new data output format, other than submitting a request for such a format to be supported by the site maintainers.

Storage Resource Broker

Storage Resource Broker (SRB) is a project based at the San Diego Supercomputing Center, whose stated goal is to

> [provide] a uniform interface for connecting to heterogeneous data resources over a network and accessing replicated data sets. SRB, in conjunction with the Metadata Catalog (MCAT), provides a way to access data sets and resources based on their attributes and/or logical names rather than their names or physical locations. [SRB05]

While there is some overlap between SRB and IDACT, one significant difference between the projects is the lack of any transformation capability for data input or output. While SRB does allow access to a variety of distributed, heterogeneous data sources, no provision is made for automatic transformations between the data source format and the format required by the data consumer, which is precisely the goal of this effort.

SOLUTION SCOPE

Intelligent Data Acquisition, Collection, and Transformation (IDACT) System

Overview

In 2003, NASA awarded an Advanced Information Systems Technology (AIST) Program grant to the University of Alaska Fairbanks to develop the IDACT System (NASA award AIST-02-0135) [Hay03, HaNa04, LNH04, NaHa04a, NaHa04b, NaHa05]. This system addresses the problems raised in Chapter 1 as increasingly complex scientific models attempt to utilize multiple, possibly geographically distributed, data sources as their inputs. IDACT provides a mechanism which allows data consumers to more easily access data from multiple heterogeneous sources, and is based on several components, as shown in Figure 3.1.

The five major IDACT components, described below, are the Data Source Registry, the Query Manager, the Data Warehouse, the Transformation Manager, and the Schedule Manger. These components function in sequence to provide data consumers with results in a form in which they can be most effectively utilized. IDACT was initially conceived for use in earth science environments, but its modular design allows it to be applicable to most data-driven domains. A given IDACT instance can involve the use of all of the components shown in Figure 3.1, but, depending on the application, some of the components, such as the Transformation Manager, can also be employed in a standalone fashion through the use of a webservice interface.

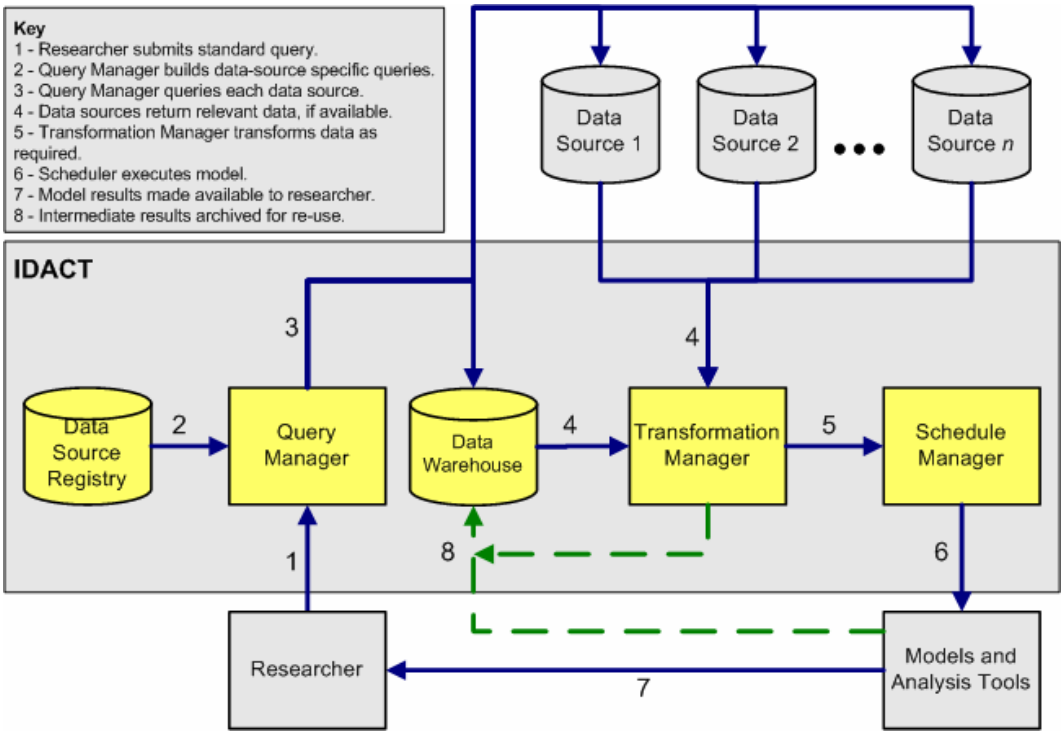

Figure 3.1: IDACT Overview [NaHa04b]

Data Source Registry

The Data Source Registry (DSR) tracks the structure and access method for multiple data sources [Cre03, Cre05]. In addition, it allows associations between components of the data sources to be defined, so that, for example, datasets which contain related data from different sources, each with their own structures, can automatically be integrated into a seamless view. Such associations can be created manually by a user or administrator, but are also created automatically as new data sources are made available. This allows even a novice user to add a new data sources to the registry and have it be available for use in the system with minimal user intervention.

Query Manager

The Query Manager (QM) accepts data requests from data consumers in a variety of formats based on the requirements of various user groups. For example, for some users, a simple *picklist* interface may be a sufficiently powerful method for requesting data, whereas other users require more detailed and descriptive query mechanisms. However, the QM serves to abstract data consumers from the underlying organizational schema of the actual data sources, allowing them instead to focus on a request that is tailored to their level of technological expertise.

Once the QM receives a request for data, it builds a series of *queries* for each of the relevant data sources, based on the information stored in the DSR. In the context of the QM, the term "query" is used in a broader sense than just as a reference to SQL statements, but instead applies it to any data search and acquisition method. As such, a particular QM query may involve a search of a directory for files, or a webservice call, for example. The query manager is under development at this time.

Transformation Manager

The results of queries generated by the QM are accepted by the Transformation Manager (TM) component [Hay03, HaNa04, LNH04, NaHa04a, NaHa04b, NaHa05], which attempts to ensure that the data is presented to the data consumer in the manner in which it is requested, rather than in the format used by the underlying data source. The TM component is the focus of this research effort, and its function is described in greater detail in the following chapters.

Schedule Manager

The Schedule Manager (SM) component allows for actions, such as the invocation of a decision support system or model, based on events such as time periods, or the acquisition of new data [Lis03, Lis05, LNH04]. Using a Petri-net based approach, the SM uses a combination of boolean and temporal operators to specify the conditions under which the defined actions will occur. A basic example of such a rule is:

Create product P using datasets A, B, C if
A and B have both been updated since P was last created, OR
Dataset C has been updated since P was last created, OR
It has been more than 8 hours since P was last created.

The triggers for a specific action can range from the very simple (e.g. trigger whenever new data is acquired) to the very complex (e.g. boolean expressions involving multiple data sources and time periods).

Data Warehouse

The Data Warehouse (DW) component is used to archive commonly requested data. Data can be submitted to the DW from the TM as a dataset after a transformation has been applied to source data by the TM, or as a data product resulting from the application of a model or analysis tool.

Solution Scope

The scope of this research effort is to identify methods by which the Transformation Manager (TM) component of IDACT can effectively automate much of the process of transforming data between data formats, particularly in cases where either the input or output format is not a recognized standard. The major deliverable of this research effort is the prototype TM implementation which utilizes these methods and an accompanying analysis of the effectiveness of the automation techniques.

SOLUTION DESCRIPTION

Overview

The issue of providing data in a format which meets the needs of the recipient is a complex one, although it has the potential to greatly improve the operation of data centers, which in turn will make data more useful and more likely to be applied to address scientific and social questions. In particular, a mechanism for performing data format transformations is applicable to the following functions in the data center environment:

- It will allow data collectors to submit data to data archivists in a far more flexible manner. The lack of such an option has been identified [NaHa01] as significantly reducing the likelihood of data collectors attempting to submit data to data centers, and of their submissions actually being accepted.
- It will allow data consumers to access data in a manner which suits their needs, rather than using a format defined by the data archivist. Allowing data to be distributed in such a manner decreases the level of technical proficiency required on the part of data consumers, resulting in a more widespread use of valuable datasets.

In this work, data is considered in two forms: the native (or source) format, and the desired (or target) format. In the case of the data collector submitting data to a data center, the source format is the format in which the submission is made, and the target format is the data storage format (or some loading format) used by the data center. In the

case of the data consumer, the source format is one defined by the data center, and the target format is the format required by the consumer, which could be the input format for an analysis tool or model, for example.

Two categories of transformation will also be considered, namely those that will impact the format of the document containing the data, and those which impact the data itself. As a simple example, it is possible to transform a dataset stored in a comma-delimited text format to an XML-based format without making any change to the data contained within the file. However, it is also possible to perform some operations on the data contained with the file, such as converting a field containing speed measurements from miles-per-hour to meters-per-second.

Transformation Mechanisms

Introduction

In order to provide a format transformation mechanism, there are essentially four cases which should be considered. These include identity transformations, predefined transformations, transformation sequences, and custom transformations. Each of these cases are included in the Transformation Manager prototype implementation. The TM maintains a description of each available transformation it is capable of performing, and searches the list in order to determine the most suitable approach to fulfilling a given transformation request.

Identity Transformations

In order to provide such a format transformation mechanism, there are essentially four cases which should be considered. The first such case is the most trivial, as it involves a source format and a target format which are identical. In such cases, the transformation simply becomes a pass-through, or identity transformation, in which the output is the same as the input.

Predefined Transformations

In some cases, primarily those which involve commonly used standard formats for the source and target, there are predefined tools which perform format transformations. For example, there are several tools which perform various conversions to and from HDF5 [NCS05b]. In cases where such tools exist, they can be applied to perform transformations which meet the needs of the user. A mechanism which transforms from between source format A and target format B will be denoted as $T_{A, B}$.

A search to determine whether a suitable predefined transformation exists within a particular IDACT instance is trivial to perform. The search of the set of all available transformations, S_T, where $|S_T| = n$ can easily be performed in O(n) time if S_T is unsorted, and O(log n) time if S_T can be accessed in a more ordered manner, such as via an index to database records, or even O(1) time if a perfect hash table is assumed to be available.

Transformation Sequences

In the event that a single predefined transformation does not exist, it may be

possible to construct a sequence of transformations which meets the goal of the user. For example, given source format *A*, and target format *B*, there may not be a transformation $T_{A, B}$. However, if intermediate format C exists, in addition to transformations $T_{A, C}$ and $T_{C, B}$, then it may be possible to effectively perform $T_{A, B}$, by applying he transformation sequence $TS_{A, B} = (T_{A, C}, T_{C, B})$. In a more general sense, a transformation sequence which transforms format 1 into format *m*, using intermediate formats [2, (*m*-1)] is given as

$$TS_{1, n} = (T_{1, 2}, T_{2, 3}, \ldots, T_{(m-2), (m-1)}, T_{(m-1), m})$$

A single transformation can essentially be viewed as a transformation sequence of length 1, which is useful in the actual implementation of the prototype Transformation Manager.

Given a set of transformations, determining the lowest cost transformation sequence between source format *A* and source format *B* is essentially equivalent to solving the shortest path problems for a graph in which the data formats are represented by vertices, and the transformations are represented by edges. Each edge in the graph can be weighted with the cost of the given transformation. Such weights may be determined in many ways, but generally will provide an estimation of the computational time or complexity of the transformation. Equal weights may also be used for all edges, in which case the lowest cost transformation sequence would equate to the sequence with the fewest steps. For example, Figure 4.1 shows an example in which there are five data formats, and six defined transformations between them. The shortest path problem can be solved by Dijkstra's algorithm, for example, for a given number of data formats, *f*, in $O(f^2)$ time.

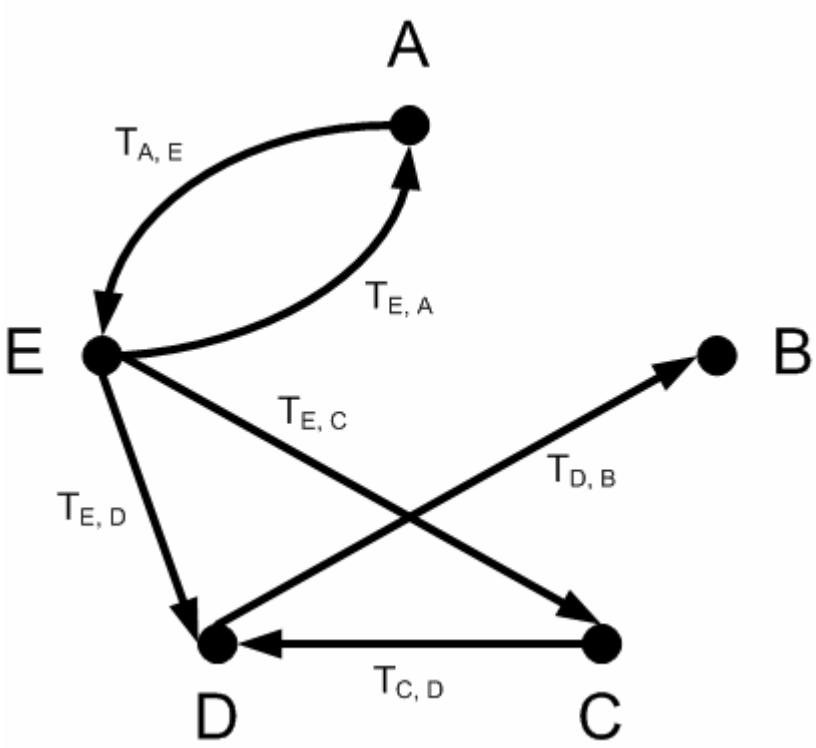

Figure 4.1: A graph representation of a set of six transformations between five formats.

While it is feasible within the current TM architecture to use non-uniform weights in the transformation graph, the problem of assigning such weights presents a significant challenge which is likely to be insurmountable in a general sense. The first issue with assigning such weights involves the choice of metrics, which could include computational time, computational space, or even a measure of the precision with which the transformation is performed. Since at least some of these values are generally dependant on the size of the input, the metrics would also have to be expressed in a manner that accounted for such differences. Assuming that the relevant metrics could be identified and expressed in a useful manner, the next issue to be addressed would be the method by which these metrics were combined to form the weights for each

transformation. For example, are computational space and computational time equally important, or is one more important than the other, and if so by what factor? Finally, it is important to recognize that the metric for a given transformation may be impacted by factors other than the size of the dataset to be processed, and the size of the fields within the dataset. Suppose, for example, that two otherwise equivalent transformations sort data, one using bubble sort and the other using quicksort. It would be easy to conclude that quicksort would be preferable, but in the case of a source dataset which was already sorted, or nearly sorted, this would generally not be the case. Likewise selection sort would be preferable to both if there are large records within the datasets to be manipulated, and the cost of a *move* operation exceeds the cost of a *comparison operation*.

While a general solution to this problem would be useful to this work, and in other domains, such as job scheduling, the act of assigning weights appears to be a problem that would need to be solved for each invocation of the TM, which does not seem to be useful, reasonable, or computationally efficient as it may require more effort to determine the weights for each transformation request than to perform the transformations. The solution used for the TM, therefore, is to assign uniform weights to all transformations, and use the shortest path as the automatically selected transformation sequence. However, the administrator of any given TM instance is free to modify those weights if they can determine suitable values for the transformations and datasets typically encountered within their domain. While not a general solution to this problem, such weights could be determined for a given domain using empirical data, based on the

application of the set of transformations to representative datasets.

A second, and somewhat related, issue in the choice of transformation sequence is that of finding several equally suitable transformation sequences, which in the case of the uniformly weighted transformations equates to transformation sequences of equal length. For example, consider the set of transformations represented by the graph shown in figure 4.2, in which the transformation sequences $\{T_{A, B}, T_{B, D}\}$ and $\{T_{A, C}, T_{C, D}\}$ are of equal length, and both sequences ultimately perform a transformation from source format A to target format D.

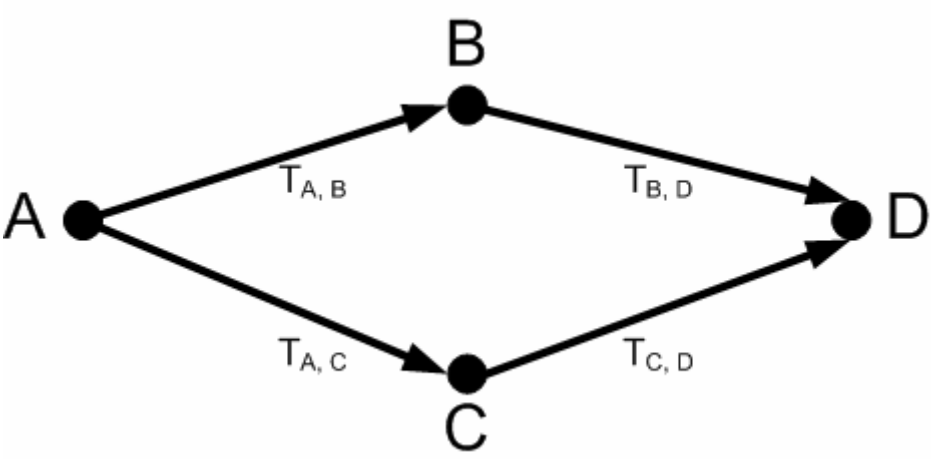

Figure 4.2: A graph representation of a situation in which there a two possible transformation sequences between source format A and target format D.

The current TM architecture addresses this issue by selecting the first suitable sequence found, and allowing the user to modify the choice of transformation sequence by presenting them with relevant alternative options, which allows a reasonable solution to be chosen automatically, which allowing the user to manually select a prefer transformation if they so desire. There are other techniques that could be applied to

address this issue, including tracking the number of times that a given transformation sequence was applied and using that value for conflict resolution. However, no matter what method is chosen, if two or more transformation sequences exist for a given transformation request, there always exists the possibility that a given user will select to use a transformation sequence other than the automatically selected one. As such, the TM follows the approach of making a reasonable choice of transformation sequence, and providing the user with the option of selecting from one of the other sequences if desired.

Custom Transformations

In the event that no suitable transformation sequence is available to meets the needs of the user, it may be possible to automatically construct a new transformation, assuming that the source and target formats are in some sense compatible (i.e. there must be some level of commonality between the formats, otherwise the concept of a transformation is meaningless). While an experienced computer programmer could construct such a custom transformation, this is often beyond the abilities of the typical data consumer, and is often not practical beyond the development of mechanisms to transform between standard formats. As such, some methods for automatically constructing such transformations are considered. If a custom transformation is created, it is not only applicable to the transformation of the current source data, but is also made available to all users for future transformations.

Custom Transformation Creation Process

The approach used in creating custom transformation begins with the association of fields in the source and target formats to common fields, as defined in the Data Source Registry (DSR) [Cre03, Cre05]. These mappings, or *associations*, are stored in a database, which is constructed and managed by the DSR, and while associations can be defined manually using the DSR interface directly, the associations in this effort are created via the TM, as a result of either automated processing or user selection.

Transformation Components

A Transformation Component (TC) describes how to perform an operation on a single data item. These operations are typically quite basic, such as converting a speed from miles-per-hour to meters-per-second, or extracting a substring from a field value. The transformation components for a given instance of the TM/DSR are stored as XSLT templates, based in part on the XSLT Standard Library [Sou05], and augmented with any additional relevant domain specific XSLT templates. In the event that a suitable TC is not defined in the system, a user can define a new TC whenever necessary, which is then added to the library and made available to all users.

At present, approximately 75 XSLT transformation components, covering operations including unit conversion and the manipulation of strings and geographic coordinates have been written as part of this work. As further application domains are identified, a domain expert will be employed to develop a set of relevant domain specific XSLT transformations for deployment with TM instances. At present, this effort is

already underway in the chemistry domain, using the skills of a chemistry domain expert. A detailed description of how the XSLT Transformation Components are applied in the generation of a new custom transformation is given on page 64.

Common Fields

The DSR manages a set of common fields, to which each of the identified data format fields are linked, through a defined association. The set of common fields is not static, and can be expanded by the user if a suitable common field does not exist. For example, if the user has a source field which is related to speed, but there are no suitable common fields to which it can be mapped, then a common field named *velocity* can be added. There is also a mechanism in the DSR by which a domain expert can validate the common fields added by the user. If, for example, the *velocity* common field added by a user contained only a speed but no direction, then the name *speed* would be more appropriate. The domain expert could then update the name of the common field, and the DSR then indicates that the common field was verified by the domain expert, which gives it more credibility than those common fields which have merely been added by system users.

To allow for the greatest degree of flexibility in mapping source fields to target fields, an attempt is made to reduce the complexity of common fields. For example, rather than having a common field named *geographic-coordinates* in which both latitude and longitude are stored, it is more effective to have two common fields named *latitude* and *longitude*.

Simple Associations

Each simple association directly maps a single field in the source format to a single common field. For example, the source field named *vehicle speed* could be associated with a common field named *speed*, and shown in Figure 4.3.

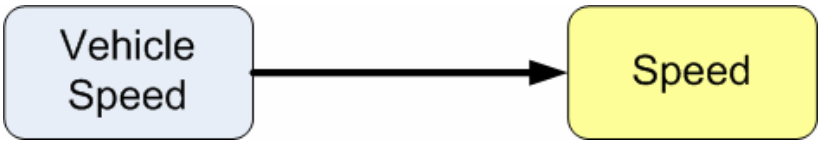

Figure 4.3: A simple association between the source field *vehicle speed* and the common field *speed*.

The association itself identifies fields which have related data. One or more transformation components can optionally be included with the association to define how the information in the source field actually relates to the common field. For example, suppose the *vehicle speed* field contains data measured in kilometers-per-hour, and the *speed* common field expresses the same information in meters-per-second. A TC is applied to the association to describe how the data should be converted from the source field form to the common field form. In this case, the TC would involve multiplying the value in the *vehicle speed* source field by 1000/3600, or 10/36, to get the equivalent value to the *speed* common field. An example of such an association can be seen in Figure 4.4.

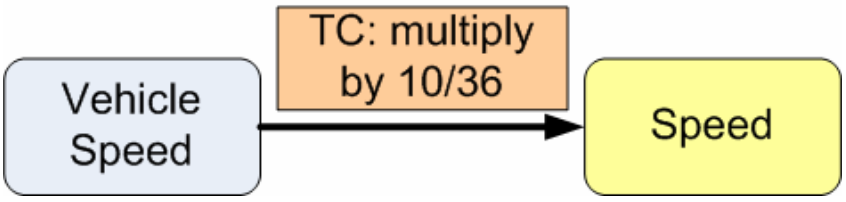

Figure 4.4: A simple association and the relevant transformation component between the source field *vehicle speed* and the common field *speed*.

Once a TC has been applied, the resulting value can be adjusted to contain approximately the same number of significant digits as the original value, in order that the common field value not appear more precise than the data on which it is based. By performing this as an independent step, rounding can be left to the end of the process in the event that several TC stages are applied to a source field value.

Split Associations

A split association can be applied to define how a single source field relates to more than one common field. For example, the source field *geolocation*, which contains a latitude and longitude, may be related to the common fields *latitude* and *longitude*, as shown in Figure 4.5.

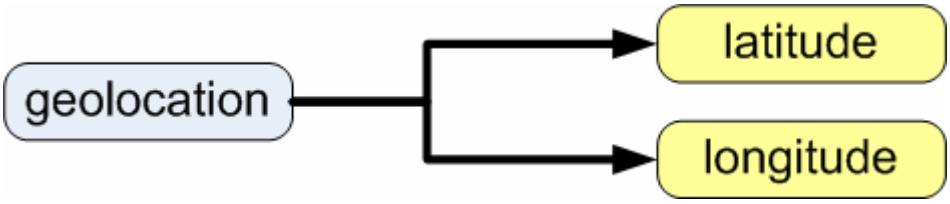

Figure 4.5: A split association between the source field *geolocation* and the common fields *latitude* and *longitude*.

As with the simple association, TCs can be included with the split association to describe how to form the common fields from the source data. For example, suppose that geolocation involves values of the form latitude; longitude using degrees, minutes, and seconds, while each of the two common fields uses decimal degrees. The association and the TCs can be represented as shown in Figure 4.6.

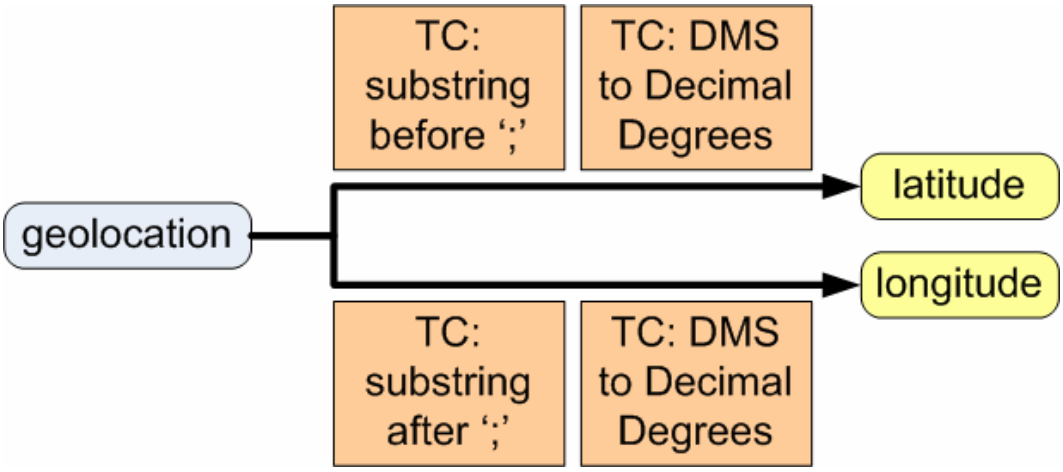

Figure 4.6: A split association and the relevant transformation components between the source field *geolocation* and the common fields *latitude* and *longitude*.

Combine Associations

A combine association can be applied to define how a multiple source fields relate to a common field. For example, a combine association may be used to relate the source fields *date* and *time-of-day* to the common field *time*, in which the time is stored as second since some epoch, such as the commonly used 0:00 on January 1st, 1970. This example is depicted in Figure 4.7.

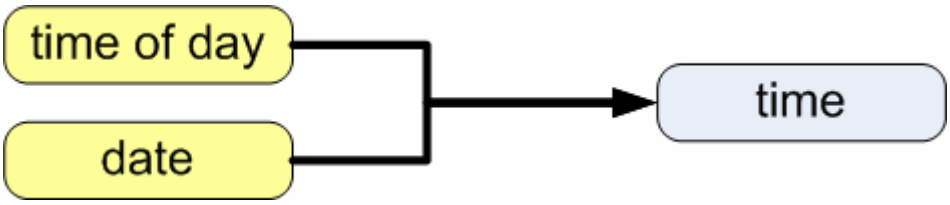

Figure 4.7: A combine association between the source fields *time-of-day* and *date*, and the common field *time*.

As with the other association types, TCs can also be included to describe the mechanics of the transformation process. Figure 4.8 shows how the TC *dateTime*-to-

timeSinceEpoch can be applied to this process.

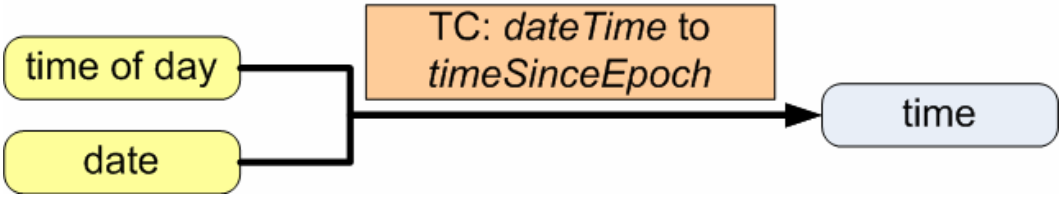

Figure 4.8: A combine association and the relevant transformation component between the source fields *time-of-day* and *date*, and the common field *time*.

Association Database

The DSR maintains a database of currently defined associations between source fields and common fields, which is organized into Owner Domains (OD), based on the identity of the user who created the association. For example, in a system with two users, Scientist A (S_A) and Scientist B (S_B), each would have their own OD, and an additional OD, named Global, would exist to contain system wide associations. Figure 4.9 shows an example of a simple DSR database, in which three owner domains and four common fields have been defined. In this example, the source field *Location* is associated to common fields in three different ways.

1. In the *Global* OD, *Location* is associated with the common fields *Latitude* and *Longitude*, in a form similar to the example in Figure 4.4.
2. In the S_A OD, *Location* is associated with the common field *City*. This association was added as a result of S_A processing data in format A.
3. In the S_B OD, *Location* is associated with the common field *Country*. This association was added as a result of S_B processing data in format B.

TC information is also stored with each of these associations, although this is not specifically indicated in the logical view shown in Figure 4.9.

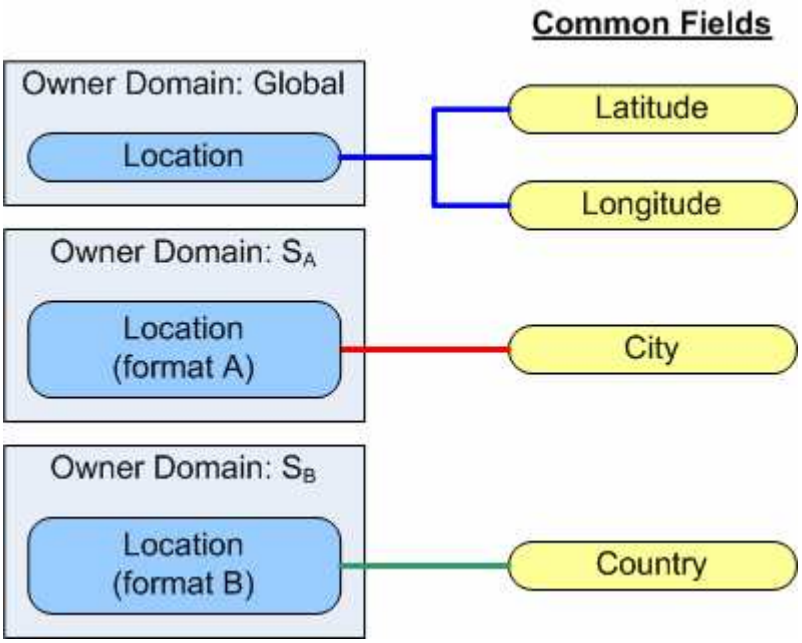

Figure 4.9: An example subset of the DSR database, in which there are three owner domains, and four common fields.

Adding New Data Sources

If a new data source format, C, is used by S_A, associations will be added to the DSR database which define how the fields in C map to the common fields. In such cases, an ordered list is automatically constructed to determine an appropriate association, and to allow the user to select from a list of likely associations should the automatically selected association not meet their needs. The process used to construct the list, using the example of S_A processing format C, in which there is a *Location* field, is as follows:

1. Determine if there are any existing associations defined in the current OD for the given field and format. Any associations found during this search are added to

the list. In this example, this would involve a search to determine if there are any associations for the *Location* field for format C in the S_A OD, which there are not.

2. Repeat step 1 for the Global OD, and append any associations found to the list.
3. Repeat step 1 for all other ODs (i.e. all ODs other than S_A and Global in this case), and append any associations found to the list.
4. Determine if there are any existing associations defined in the current OD for the given field name, and append any associations found to the list. In this example, this would involve a search to determine if there are any associations for the *Location* field for any format in the S_A OD. One such association exists, so the list would become {*City*}.
5. Repeat step 4 for the Global OD. In this example, one such association exists, so the list would become {*City*, (*Latitude*, *Longitude*)}.
6. Repeat step 4 for all other ODs. In this example, one such association exists, so the list would become {*City*, (*Latitude*, *Longitude*), *Country*}.

At the end of this process, the first item in the list is used as the automatically selected association, and the process continues for other fields in the new data source. If the user accepts the automatically selected association, then the database is updated with the new association. For example, if S_A accepts the association of *Location* to *City* for source format C, then the database is updated as shown in Figure 4.10.

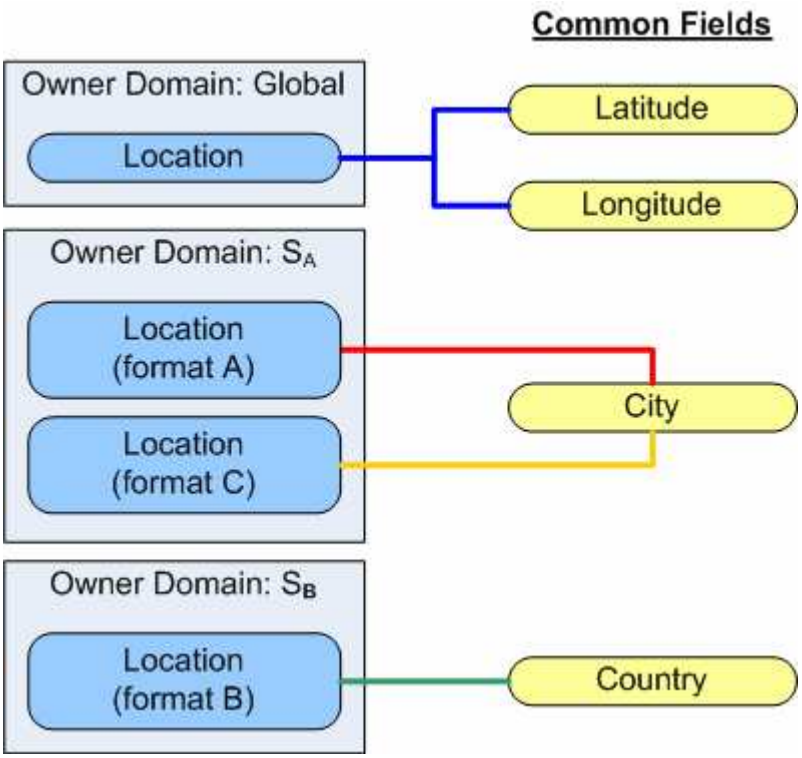

Figure 4.10: An example subset of the DSR database, after S_A accepts the proposed association between the *Location* field from format C and the *City* common field.

Given the database depicted in Figure 4.10, if S_A attempts to create an association for the *Location* field from format D the resulting list will again be {*City*, (*Latitude*, *Longitude*), *Country*}. However, in this case, suppose that S_A rejects the automatically selected association, and determines that associating *Location* with *Country* is more appropriate. The database would then be updated as shown in Figure 4.11.

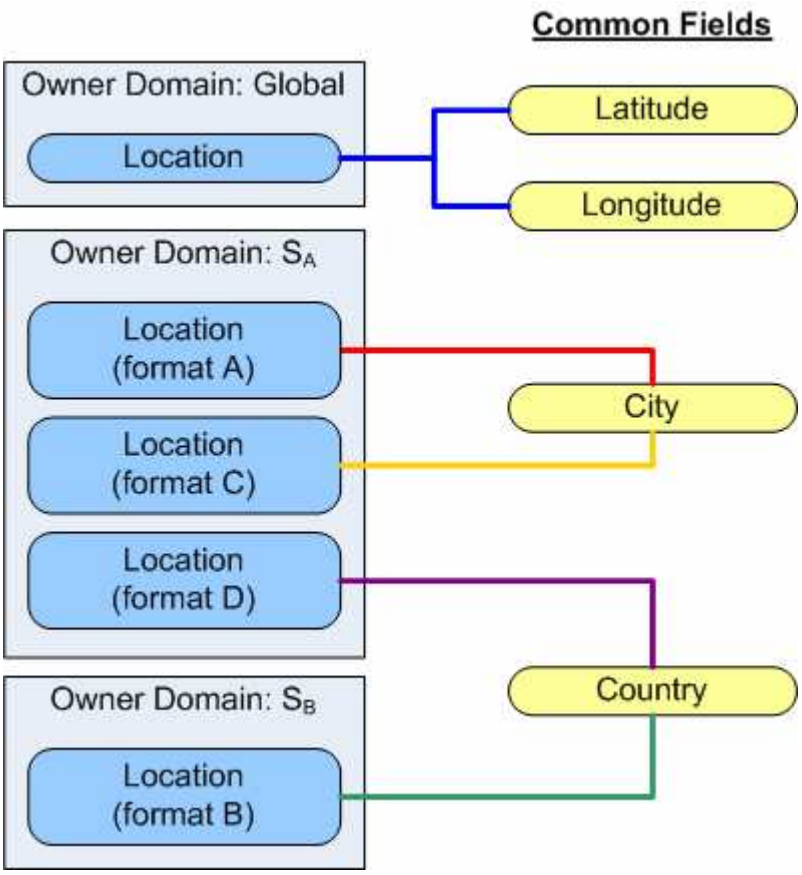

Figure 4.11: An example subset of the DSR database, after S_A rejects the proposed association between the *Location* field from format D and the *City* common field, and instead chooses to associate the *Location* field with the *Country* common field.

Conflict Resolution in Building the Candidate Association List

When attempting to build the list of candidate solutions, as described on page 41, it is possible that at any of the steps a conflict may arise, in that more than one possible association may be found. For example, if S_A attempted to create an association for the field *Location* from format E given the DSR database state depicted in Figure 4.11, then there would be two candidate associations (*City* and *Country*) found in step 4 of the

process. The mechanism to resolve this conflict and place the associations in the candidate list in order is as follows:

1. Preference is first given based on the context of the fields. This is not possible in all cases, but the mechanisms for performing context-based conflict resolution are addressed on page 46.
2. If a context-based approach does not resolve the conflict, then preference is given to the association which appears as a result in the current search most frequently.
3. If frequency cannot resolve the conflict, then preference is given to the association which was most recently added.

Using the example of building the candidate association list for S_A adding the *Location* field from format E, the conflict would be resolved using frequency. In the S_A OD, there are two associations to *City*, and only one to *Country*, so the associations would be added to the list in the order {*City*, *Country*}. The final list, after the remaining steps were completed, would be {*City*, *Country*, (*Latitude*, *Longitude*)}.

Partial String Matching for Name-Based Field Mapping

The process used to build the ordered list of candidate associations as presented relies on exactly matching a new field name with one of the field names in the DSR database. However, it is likely that new field names may be similar to existing names, while not necessarily matching exactly, and it is also important to identify these partial matches as candidate associations. Based on a similar approach used by SIMON [HaNa00, Hay00], steps 4-6 defined on page 42 can be repeated using substring matching

(new field name as a substring of an existing name, and vice versa) to generate additional entries to be appended to the candidate association list.

Given the example DSR database depicted in Figure 4.11, the resulting candidate association list would be empty for a field named *Event Location* using only exact matching. However, when followed by a substring matching process, the candidate association list would include several items, as *Location* is a substring of *Event Location*.

Context-Based Conflict Resolution

In the case of hierarchically organized data formats, such as XML-based formats, it is often possible to use context as a method for conflict resolution, or to determine a set of likely associations in the event that the other methods have failed. In such cases, the context is defined as the path from the root of the document to the current field. Consider the example of determining an association for the field *Minutes* in a new data format *D*, given the subset of the DSR database depicted in Figure 4.12.

In this example, it is not clear which association would be most appropriate for the new field *Minutes*, as there are two possible choices based on the currently defined combine associations. However, if field under consideration is viewed in context, as shown in Figure 4.13, a determination can be made as to which association is more appropriate, based on the association of the parent field *Lat* with the *Latitude* common field in data format A.

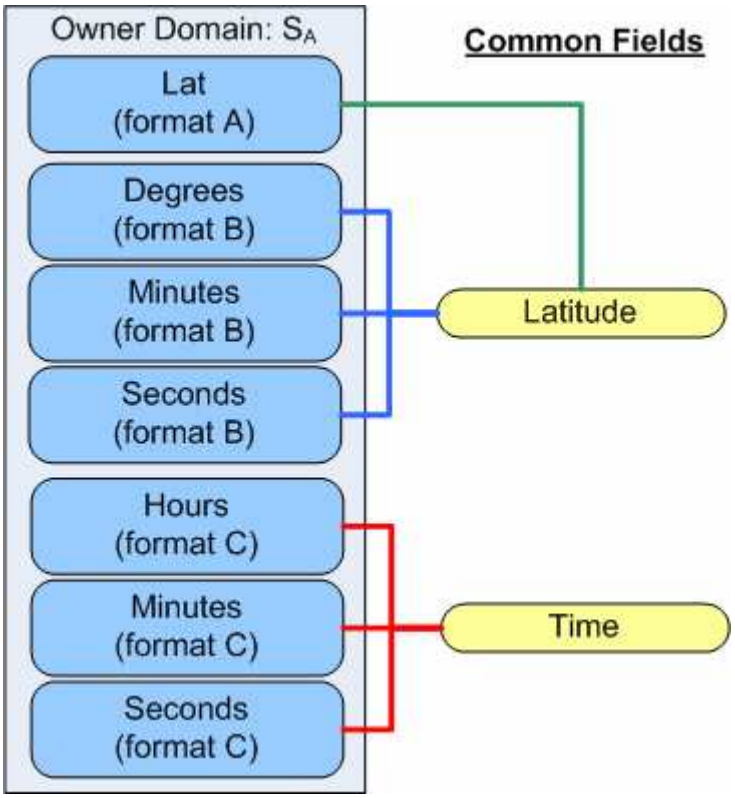

Figure 4.12: An example subset of the S_A OD in the DSR database.

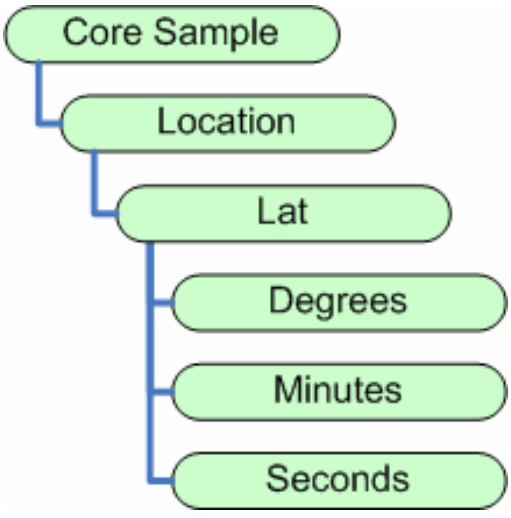

Figure 4.13: A hierarchical view of part of dataset *E*, in which the *Minutes* field is placed in context.

Creation of New Custom Transformations

Given two formats for which associations to common fields have been defined, it may be possible to build a new transformation. Consider the example of a source data format corresponding to the XML schema shown in Figure 4.14, and a target data format corresponding to the XML schema shown in Figure 4.15.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:complexType name="SENSOR_TYPE">
    <xsd:sequence>
      <xsd:element name="SENSOR_NAME" type="xsd:string" minOccurs="1" maxOccurs="1" />
      <xsd:element name="SENSOR_LOC" type="xsd:string" minOccurs="1" maxOccurs="1" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="MEASUREMENT_TYPE">
    <xsd:sequence>
      <xsd:element name="SENSOR" type="SENSOR_TYPE"
        minOccurs="1" maxOccurs="1" />
      <xsd:element name="WEATHER" type="WEATHER_TYPE"
        minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="WEATHER_TYPE">
    <xsd:sequence>
      <xsd:element name="TIME" type="xsd:dateTime" minOccurs="1" maxOccurs="1" />
      <xsd:element name="TYPE" type="xsd:string" minOccurs="1" maxOccurs="1" />
      <xsd:element name="VALUE" type="xsd:integer" minOccurs="1" maxOccurs="1" />
      <xsd:element name="UNITS" type="xsd:string" minOccurs="1" maxOccurs="1" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="MEASUREMENTS">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="MEASUREMENT" type="MEASUREMENT_TYPE"
          minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

</xsd:schema>
```

Figure 4.14: XML Schema description of example source format [NaHa05].

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:complexType name="MEASUREMENT_TYPE">
        <xsd:sequence>
            <xsd:element name="LAT" type="xsd:decimal"
                minOccurs="1" maxOccurs="1" />
            <xsd:element name="LON" type="xsd:decimal"
                minOccurs="1" maxOccurs="1" />
            <xsd:element name="TIME" type="xsd:dateTime"
                minOccurs="1" maxOccurs="1" />
            <xsd:element name="SPEED" type="xsd:string"
                minOccurs="1" maxOccurs="1" />
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="WINDSPEEDS_TYPE">
       <xsd:sequence>
           <xsd:element name="MEASUREMENT" type="MEASUREMENT_TYPE"
               minOccurs="0" maxOccurs="unbounded">
           </xsd:element>
       </xsd:sequence>
    </xsd:complexType>

    <xsd:element name="WINDSPEEDS" type="WINDSPEEDS_TYPE" />

</xsd:schema>
```

Figure 4.15: XML Schema description of example target format [NaHa05].

These formats can be represented graphically as shown in Figure 4.16.

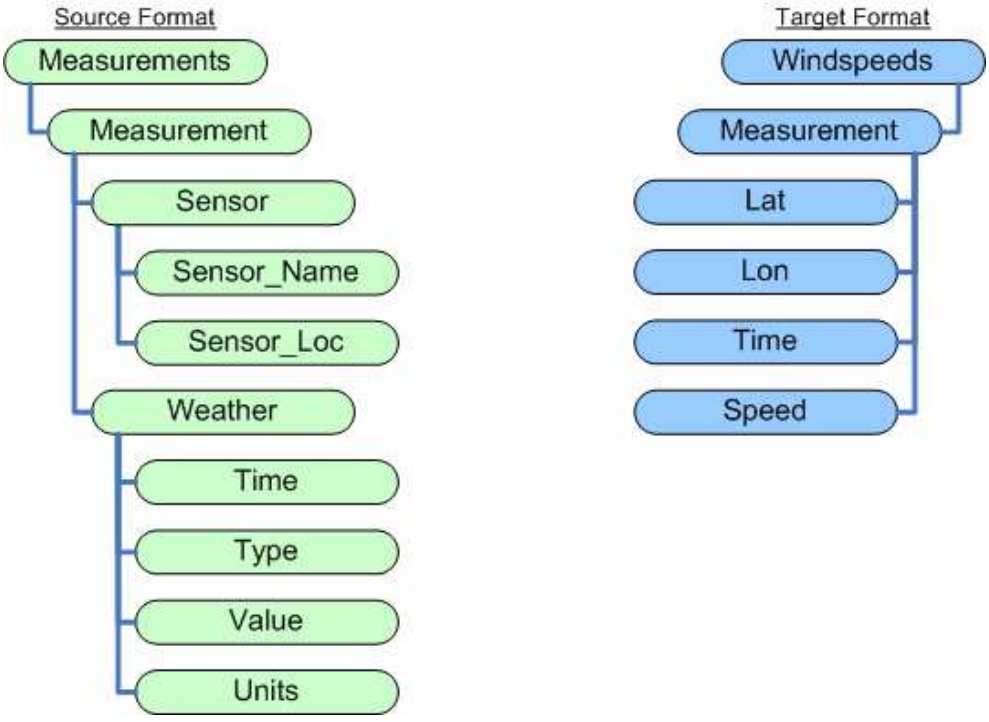

Figure 4.16: Graphical representation of example source and target formats [NaHa05].

Using the techniques previously outlined to identify associations, fields in the source format can be associated with a corresponding field in the target format, as shown in Figure 4.17.

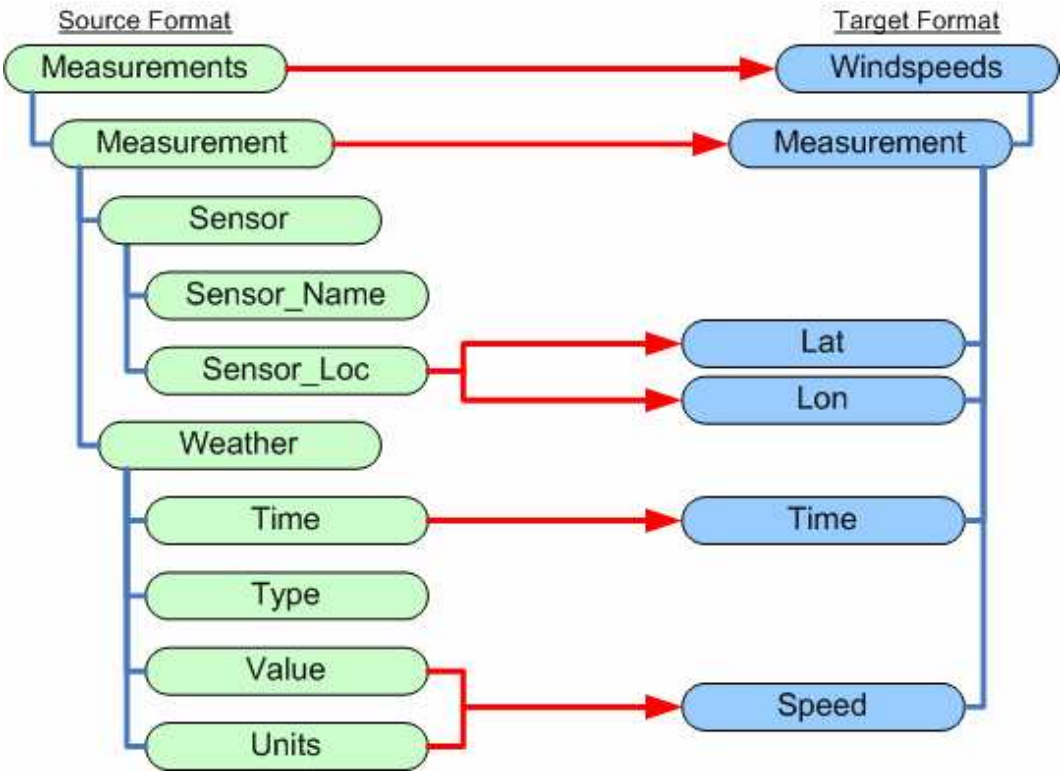

Figure 4.17: Graphical representation of associations between fields in the source and target formats [NaHa05].

If necessary, transformation components can then be added to each of the associations to define how the data from each field should be manipulated. The selected TCs are shown in Figure 4.18.

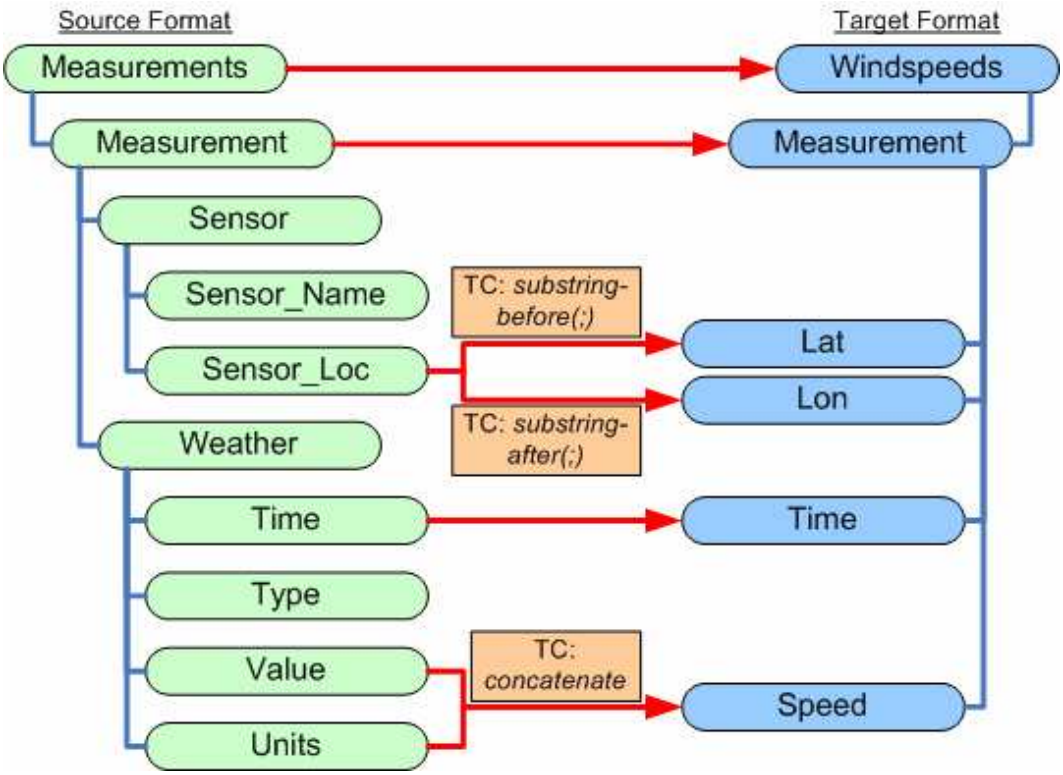

Figure 4.18: Transformation components applied to selected associations [NaHa05].

In addition, selection and sorting conditions can be added to the transformation to produce a target dataset which most closely meets the needs of the use. If, for example, the source dataset typically contained reading from several types of weather sensors, but the data consumer only required datasets related to windspeed, then a selection condition could be imposed as part of the transformation. An example of a selection condition is shown in Figure 4.19.

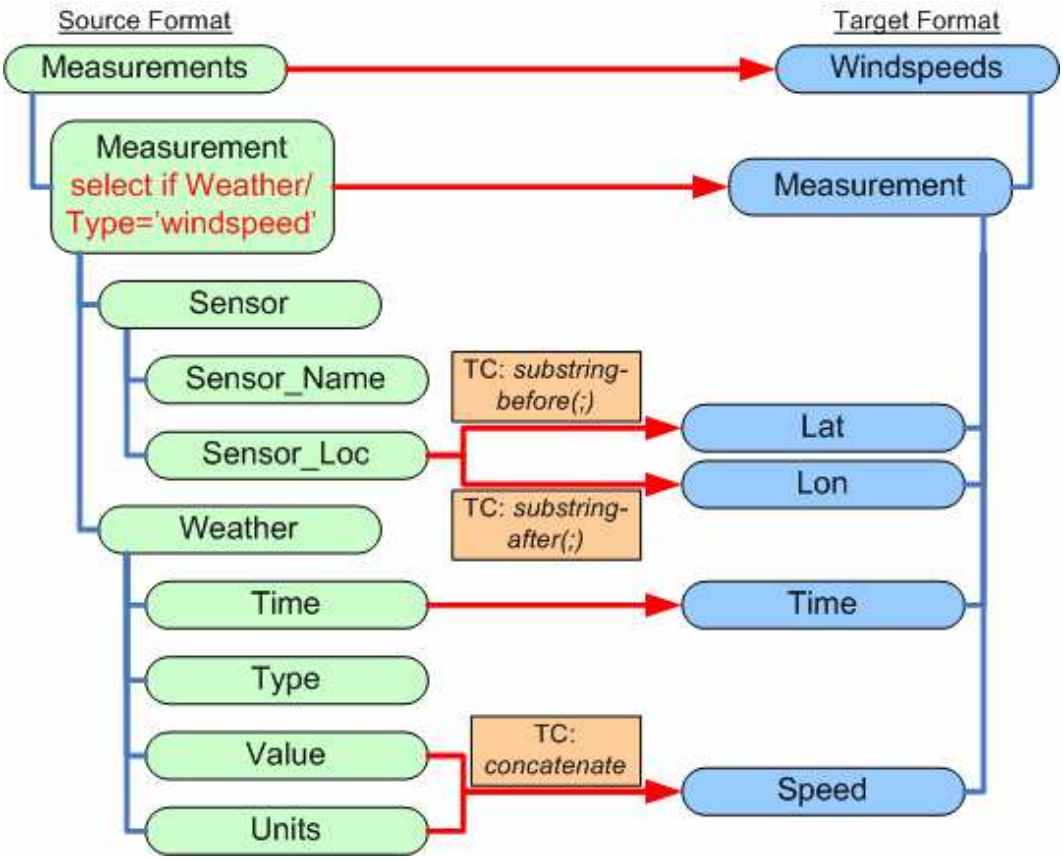

Figure 4.19: Selection conditions applied to a new transformation [NaHa05].

Using this information, the TM builds a transformation, such as the XSLT transformation shown in Figure 4.20.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" />
  <xsl:template match="/">
    <xsl:apply-templates select="MEASUREMENTS" />
  </xsl:template>
  <xsl:template match="MEASUREMENTS" >
    <xsl:element name="WINDSPEEDS" >
      <xsl:apply-templates select="MEASUREMENT">
        <xsl:sort select="WEATHER/TIME" order="ascending"  />
      </xsl:apply-templates>
    </xsl:element>
  </xsl:template>
  <xsl:template match="MEASUREMENT" >
    <xsl:if test="./WEATHER/TYPE='windspeed'">
      <xsl:element name="MEASUREMENT" >
        <xsl:call-template name="SENSOR_LOC_TO_LAT" >
          <xsl:with-param name="value1" select="SENSOR/SENSOR_LOC" />
        </xsl:call-template>
        <xsl:call-template name="SENSOR_LOC_TO_LON" >
          <xsl:with-param name="value1" select="SENSOR/SENSOR_LOC" />
        </xsl:call-template>
        <xsl:apply-templates select="WEATHER/TIME" />
        <xsl:apply-templates select="WEATHER/VALUE" />
      </xsl:element>
    </xsl:if>
  </xsl:template>
  <xsl:template name="SENSOR_LOC_TO_LAT" >
    <xsl:param name="value1" />
    <xsl:element name="LAT" >
      <xsl:value-of select="substring-before($value1,':')" />
    </xsl:element>
  </xsl:template>
  <xsl:template name="SENSOR_LOC_TO_LON" >
    <xsl:param name="value1" />
    <xsl:element name="LON" >
      <xsl:value-of select="substring-after($value1,':')" />
    </xsl:element>
  </xsl:template>
  <xsl:template match="TIME" >
    <xsl:element name="TIME" >
      <xsl:value-of select="." />
    </xsl:element>
  </xsl:template>
  <xsl:template match="VALUE" >
    <xsl:element name="SPEED" >
      <xsl:value-of select="." />
      <xsl:value-of select="../UNITS" />
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

Figure 4.20: Example XSLT transformation created by TM [NaHa05].

This transformation can then be applied to a dataset which is consistent with the source format defined by the XML Schema given in Figure 4.14, with the result being a new dataset which is consistent with the target format defined by the XML Schema given in Figure 4.15. Examples of such datasets are shown in Figures 4.21 and 4.22 respectively.

```
<?xml version="1.0" encoding="UTF-8" ?>

<MEASUREMENTS>
    <MEASUREMENT>
        <SENSOR>
            <SENSOR_NAME>Sensor
A</SENSOR_NAME>
            <SENSOR_LOC>64.8:-
147.9</SENSOR_LOC>
        </SENSOR>
        <WEATHER>
            <TIME>2004-01-02T13:20:00-
09:00</TIME>
            <TYPE>temperature</TYPE>
            <VALUE>-18</VALUE>
            <UNITS>Celcius</UNITS>
        </WEATHER>
    </MEASUREMENT>
    <MEASUREMENT>
        <SENSOR>
            <SENSOR_NAME>Sensor
B</SENSOR_NAME>
            <SENSOR_LOC>64.2:-
148.0</SENSOR_LOC>
        </SENSOR>
        <WEATHER>
            <TIME>2004-01-02T13:30:00-
09:00</TIME>
            <TYPE>windspeed</TYPE>
            <VALUE>29</VALUE>
            <UNITS>mph</UNITS>
        </WEATHER>
    </MEASUREMENT>
    <MEASUREMENT>
        <SENSOR>
            <SENSOR_NAME>Sensor
C</SENSOR_NAME>
            <SENSOR_LOC>65.3:-
148.1</SENSOR_LOC>
        </SENSOR>
        <WEATHER>
            <TIME>2004-01-02T14:00:00-
09:00</TIME>
            <TYPE>windspeed</TYPE>
```

Figure 4.21: Example source dataset [NaHa05].

```
<?xml version="1.0" encoding="UTF-8"?>
<WINDSPEEDS>
    <MEASUREMENT>
        <LAT>64.8</LAT>
        <LON>-147.9</LON>
        <TIME>2004-01-02T13:25:00-09:00</TIME>
        <SPEED>25mph</SPEED>
    </MEASUREMENT>
    <MEASUREMENT>
        <LAT>64.2</LAT>
        <LON>-148.0</LON>
        <TIME>2004-01-02T13:30:00-09:00</TIME>
        <SPEED>29mph</SPEED>
    </MEASUREMENT>
    <MEASUREMENT>
        <LAT>65.3</LAT>
        <LON>-148.1</LON>
        <TIME>2004-01-02T14:00:00-09:00</TIME>
        <SPEED>60mph</SPEED>
    </MEASUREMENT>
</WINDSPEEDS>
```

Figure 4.22: Example target dataset [NaHa05].

Conclusions

This chapter presents, at a conceptual level, the mechanisms which allow the Transformation Manager to function, including the identity transformations, existing transformation tools, transformation sequences, and the custom transformation creation process. Chapter 5 describes the use of the concepts in the implementation of the Transformation Manager prototype.

PROTOTYPE IMPLEMENTATION

Architecture

The implementation of the Transformation Manager (TM) prototype involves a multi-tiered architecture, as shown in Figure 5.1. Using such a loosely coupled approach not only allows each of the components to be developed independently, but also ensures that the user interface is not dependant on the remainder of the TM, allowing several versions to be developed in response to the needs of each particular user group.

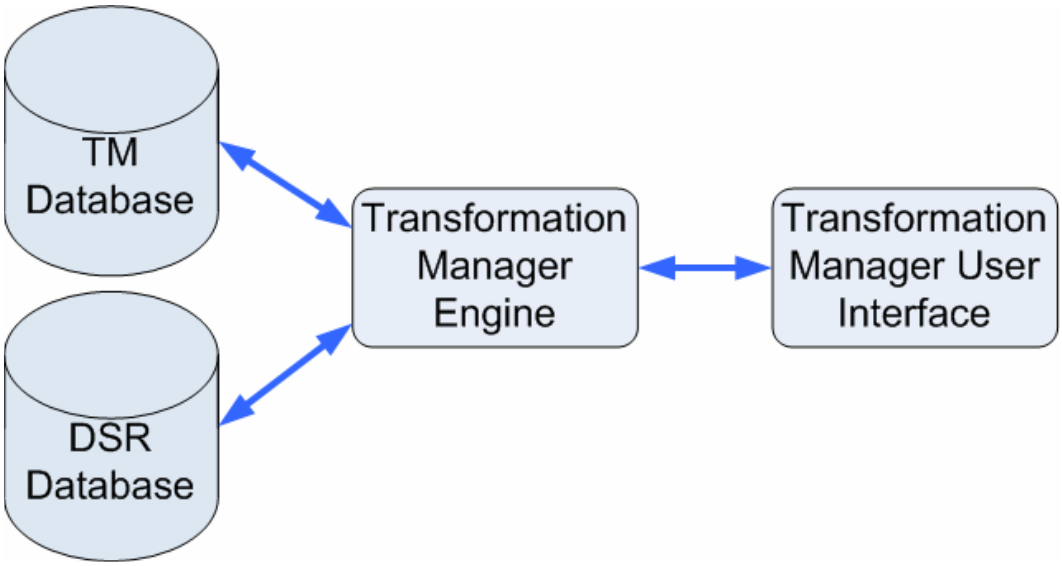

Figure 5.1: The multi-tiered architecture of the Transformation Manager.

DSR Database

The basic implementation of the DSR database was developed as an independent module during the IDACT project, using a MySQL database [MyS05] running on Linux, with a web-based UI written in PHP [PHP05], and a web-services API [Cre03, Cre05].

However, some modification was made to the DSR database as part of this effort, in order to meet the needs of the TM component. These modifications primarily involved the additional of new queries to provide the TM Engine with more effective search capabilities, particularly during the process of creating new mappings to common fields. The DSR database stores the defined associations, and also stores the transformation component (TC) library. The detailed general schema used for the DSR database is described in the DSR Technical Specification. [Cre03].

Transformation Manager Database

The TM database was also implemented using MySQL, although several other RDBMS products could have been used, requiring only a minimal change to the database driver used by the TM Engine. In practice, the TM database utilized the same database server as the DSR database in order to limit the amount of hardware required for testing, although there is no requirement for this to be the case.

The TM database is used to store information related to the currently defined transformations, allowing the TM to determine if a suitable transformation sequence exists for a given source to target format transformation. The basic schema used for the TM database is shown in Table 5.1.

TABLE: TRANSFORMATIONS		
Field Name	**Field Description**	**Example Data**
ID	The unique ID for this transformation	1
SRC_FORMAT	The name of the source format.	HDF5, weather.dtd
SRC_FORMAT_DESC	A description of the source format.	The NCSA HDF5 format, An XML weather sensor format.
TGT_FORMAT	The name of the target format.	GIF, windspeeds.dtd
TGT_FORMAT_DESC	A description of the target format.	An image in the GIF format, An XML windspeed measurement format.
TRANSFORM_TYPE	One of the several types of transformation that the TM is capable of performing.	EXEC, XSLT
TRANSFORM_NAME	The name assigned to the transformation. This is most relevant when 3rd party transformation tools are used, in which case this field contains the name of the tool.	h52gif, weatherToWindspeed
TRANSFORM_STR	A string used to invoke the transformation (for example, the path to an exectuable, or to an XSLT file).	/usr/local/bin/h52gif, dtm/trans/weatherToWindspeed.xsl
TRANSFORM_DESC	A description of what the given transformation does.	Converts an HDF5 file to a GIF file, Selects the windspeed data from a weather sensor file.

Table 5.1: Transformation Manager database schema.

Transformation Manager Engine

The TM engine accepts a transformation request, which it then attempts to satisfy if possible. The basic format of the transformation request is shown in Table 5.2.

Parameter	**Description**
Source Format	The source data format.
Source Location	The location of the source data (e.g. file, database query)
Target Format	The target data format.
Target Location	The location to which the target data should be sent (e.g. file, database insert)
Actions	A list of actions, which can be associated with a particular transformation result.

Table 5.2: Transformation request format.

Upon receiving a request, the operation of the TM is described as a flowchart in Figure 5.2

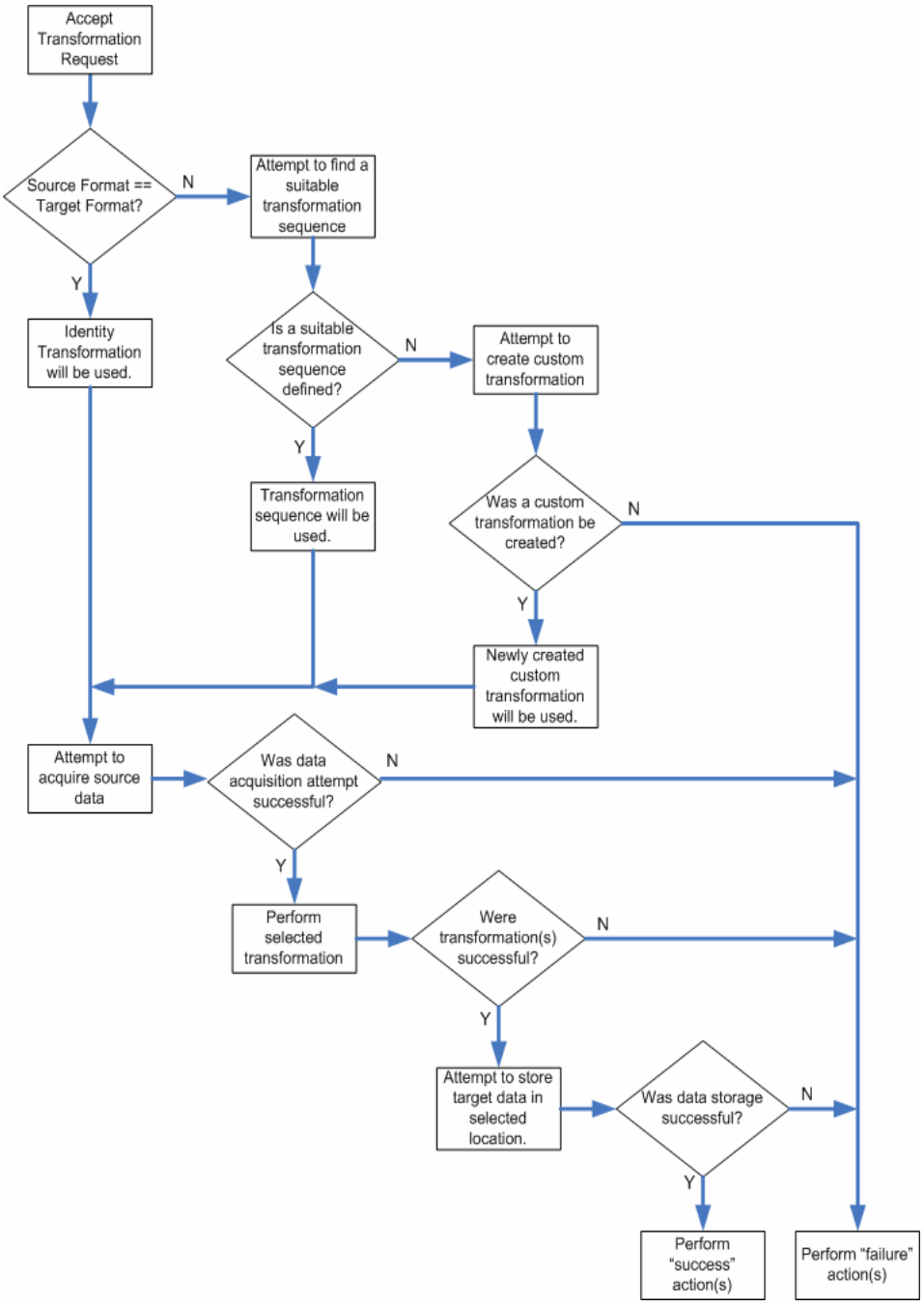

Figure 5.2: Flowchart describing the high-level operation of the TM Engine.

Implementation Tools and Languages

The TM Engine is implemented in Java [Sun05], in large part due to its seamless portability between commonly encountered systems, and because it provides a large predefined library of objects for critical TM Engine functionality, such as database interaction and XML and XSLT processing. In particular, the Apache Foundation Xerces XML Parser [Apa05] and Xalan XSLT Processor [Xal05] were used extensively via their Java APIs. However, similar tools exist for more computationally efficient languages such as C++, and future iterations of the TM could be written in such languages if improved performance of the TM Engine itself (as opposed to the transformations) is a requirement.

The current TM implementation is capable of using transformations written in many forms, including executable transformations written in any language supported by the underlying environment (e.g. compiled executables, perl, shell scripts), in addition to XSLT via an XSLT processor such as Xalan.

It is important to note that while the core TM Engine is written in Java, the bulk of the work is performed by the transformations invoked by the TM, which can be written in any language. For example, while Java may not be the best language for transformations involving large volumes of string processing, the transformation could easily be written in a more suitable language, such as C, without rewriting any of the code for the TM Engine. As such, a program (the TM Engine) which at first glance appears to be Java based, and which may therefore be considered ill-suited to high performance processing, is actually using Java for the management effort, while the

transformation instructions can be written in a form appropriate to achieving maximum performance from the hardware.

Identity Transformations

The first step performed by the TM identity transformation is to determine whether the source data is already in the form required by the data consumer, and if so an identity transformation is applied, which essentially copies the data from the source to the target location. The inclusion of identity transformations, which is essentially a copy operation, allows the TM to be more easily integrated into the IDACT infrastructure.

Internally in the TM, the transformation sequence to be applied is represented as a vector of transformation objects. In this case of the identity transformation the vector is empty, which allows the TM to process the data with no need for special logic beyond a determination that no transformations are required.

Transformation Sequences

If it is determined that the identity transformation will not suffice, the next step taken by the TM is to search the TM database for a suitable transformation sequence (TS), using the previously defined transformations. In the current implementation, the TS with the shortest number of transformations is selected by repeatedly issuing SQL queries until either a suitable TS is found, or the sequence length reaches some defined maximum length.

Suppose that an abbreviated version of the TRANSFORMATION table in the

DTM database contains three transformations, as shown in Table 5.3.

ID	SRC_FORMAT	TGT_FORMAT	TRANSFORM_DESC	Other Fields...
1	A	B	Example transformation from format A to format B	...
2	B	C	Example transformation from format B to format C	...
3	C	D	Example transformation from format C to format D	...

Table 5.3: Example records from the TRANSFORMATION table in the TM database.

In the case that of a transformation request with source format A and target format D, the first SQL query would attempt to locate a single record which satisfied those conditions. The basic structure of that SQL query is as follows:

```
SELECT
        T_0.ID
FROM
        TRANSFORMATIONS AS T_0
WHERE
        T_0.SRC_FORMAT='A' AND
        T_0.TGT_FORMAT='D'
```

This query returns an empty recordset because no single record meets the given criteria. The next query to be performed would then attempt to find a sequence of length 2, using the following SQL query structure:

```
SELECT
        T_0.ID, T_1.ID
FROM
        TRANSFORMATIONS AS T_0
        JOIN
                TRANSFORMATIONS AS T_1
        ON
                T_0.TGT_FORMAT= T_1.SRC_FORMAT
WHERE
        T _0.SRC_FORMAT='A' AND
        T _1.TGT_FORMAT='D'
```

This query also returns an empty recordset because there is no suitable transformation sequence of length 2 that meets the given criteria. This process continues to increase the length of the search sequence, and in this case the next query, shown below, would be successful in finding a suitable transformation sequence of length 3

```
SELECT
        T_0.ID, T_1.ID, T_2.ID
FROM
        TRANSFORMATIONS AS T_0
        JOIN  (
                TRANSFORMATIONS AS T_1
                JOIN
                        TRANSFORMATIONS AS T_2
                ON
                        T_1.TGT_FORMAT= T_2.SRC_FORMAT
                )
        ON
                T_0.TGT_FORMAT= T_1.SRC_FORMAT
WHERE
        T _0.SRC_FORMAT='A' AND
        T _2.TGT_FORMAT='D'
```

If a suitable transformation sequence is found, the vector of transformation objects in the TM implementation is populated with the resulting transformations.

An alternative to selecting the shortest transformation sequence is to assign weights to each of the transformations, and select the transformation sequence with the lowest total weight. A new field named could be added to the TRANSFORMATION table, and weights could be assigned to each transformation based on the computation expense of the transformation process, for example. The most suitable transformation sequence (i.e. the one with the lowest weight) could then be determined by applying Dijkstra's algorithm to the graph representation of the transformations.

Creating Custom Transformations

In the case where no suitable transformation sequence is found, an attempt is made to create a custom transformation for the user, which will then become an entity in the Transformation Library, and can be used in future transformation sequences. In the prototype TM implementation, these custom transformations are described using XSLT, which are built in part from a library of XSLT components.

The description of the source and target formats are maintained internally in the TM using *StructNode* objects. Each field is represented by a *StructNode* object, which stores the name of the field, a link to the parent node, and a vector of links to any child nodes, as shown in Figure 5.3.

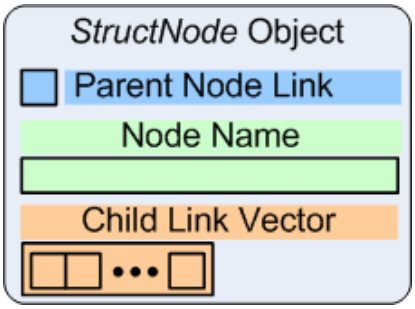

Figure 5.3: Depiction a *StructNode* object.

The data format defined in Figure 4.15 would be represented by the TM using several *StructNode* objects, as shown in Figure 5.4.

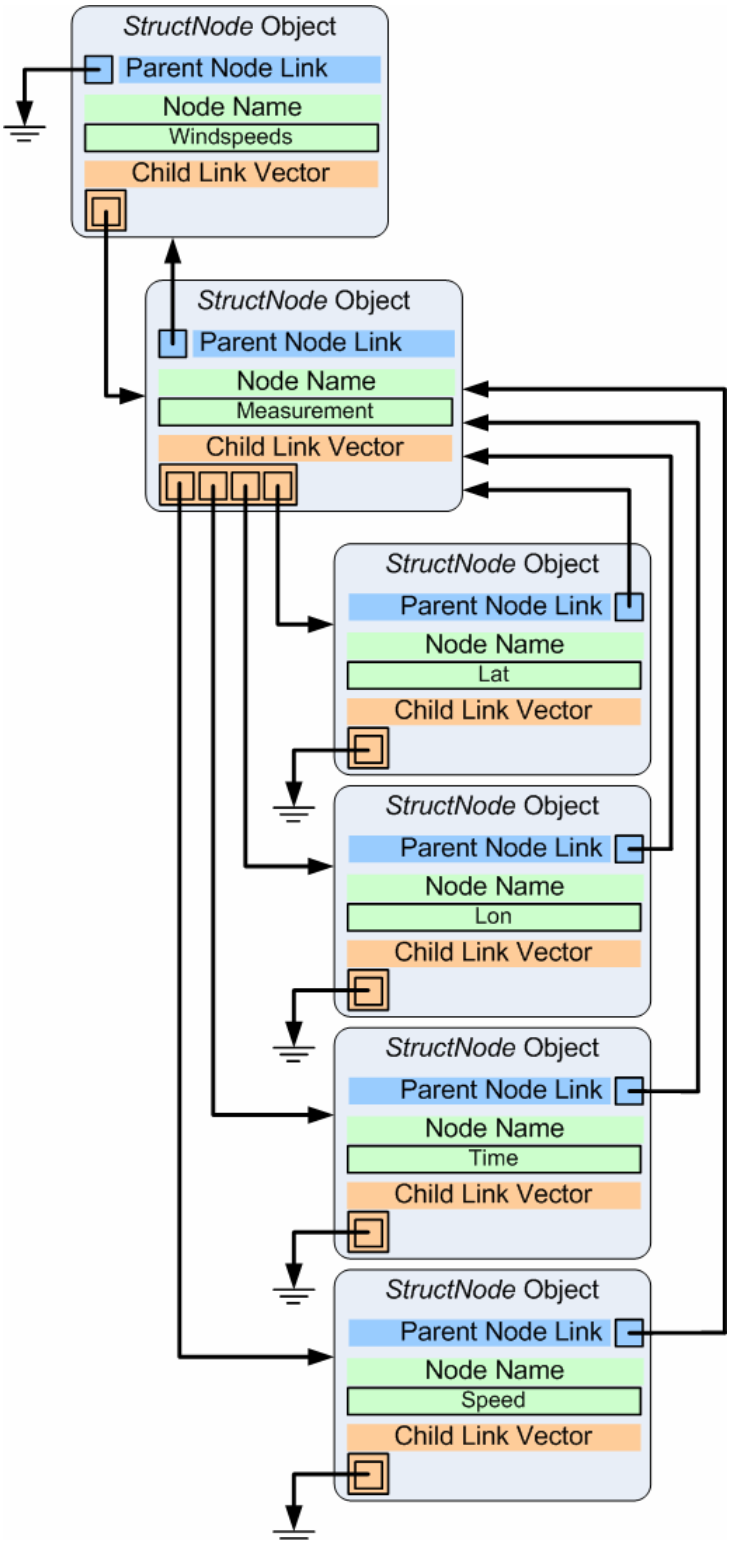

Figure 5.4: Logical representation of the target format shown in Figure 4.15.

The *StructNode* objects also include an additional element, which is a reference to a *StructLink* object, but it is only used in the case of a target field.

The TM implementation attempts to identify the associations between the fields in the source and target formats, represented by *StructNode* objects, primarily by querying the DSR database for known associations as described on page 35, using a Java Vector as the underlying data structure for the candidate association list. Once the TM has automatically identified these candidate associations, the first item in the list is selected by the TM as the proposed association. The user is then given the option to modify the associations if necessary, using either another automatically identified association from the candidate association list built by the TM, or by creating an entirely new association.

Once the associations have been defined, the related transformation components (TCs) must be identified. As with associations themselves, a list of likely TCs is determined through a search of the DSR database, after which the user is free to modify the TC selection in a manner similar to the modification of associations. The TCs themselves are defined in a library of XSLT components, which is based on the XSLT standard library [Sou05], and augmented with predefined, domain-specific TCs for the environment in which the TM is operating. In addition, a user can define a new TC, which is added to the Transformation Library and made available for future use by all TM users within the IDACT instance.

XSLT can be used to perform many types of operations, including string processing and mathematical operations, and calls can even be made to Java library routines for more advanced functionality. As a result, a given XSLT document can

describe a wide variety of transformations, which can include features such as graphical output, statistical processing, selection, sorting, and manipulation of data/time, geolocation, and measurement fields. By encoding and storing operations as XSLT components, a given operation such as unit conversion can be utilized in several transformations built from these components without the need for it to be rewritten each time, in a manner similar to the concept of object reuse in the development of programs in an Object Oriented environment.

The reuse of XSLT components also has advantages in the event that a transformation component requires modification. This can occur as a result of an incorrect definition, although the TM provides for the review of new transformation components by a domain expert in an effort to ensure that user defined transformation components are neither inappropriate nor incorrect. However, the need to modify a transformation component can also arise from an external change, which must then be reflected in the TM transformations. Examples of such events include the improvement of a statistical operation to take advantage of new techniques, or the redefinition of geographic coordinate systems [GDA05]. In such cases, it is important to update the transformation component for use in building future custom transformations. However, it may also be necessary to update any previously defined custom transformation which is based on the modified TC, although this cannot merely be implemented as replacement of the old method with the new one. In fact, there are two cases which can be applied, as follows:

1. The previously defined transformations remain unchanged, but new

transformations in which the new TC is used are generated to replace them. The previously defined transformation are then still available for use if a given user desires, but preference can be given to the newly generated transformations through the use of an increased transformation weight for the old transformation, for example.

2. The previously defined transformations are replaced with newly generated transformations, which utilize the new TC. In this case, the previously defined transformations will no longer be available for use.

The choice of which approach to apply is a dependant on the domain in which the TM is operating, and the type of modification that is made to the TC, and is left to the discretion of the TM administrator. However, the process of performing either of the two update operations described above is quite simple, as transformations components have unique names within a TM instance, and as such it is trivial to script the search for the modified TCs in the set of custom transformations. This method for ensuring that updated processes are propagated to transformation tools is more effective than that commonly applied to pre-existing transformation tools, which often rely on the user to realize that an updated software release is available, then download and install it themselves.

In the TM implementation, associations between fields, and the related TCs, are maintained using an object of the *StructLink* class. Objects of that class store contain a link to the target field object, a vector of links to one or more source field objects, and a vector of links to one or more TC objects, as shown in Figure 5.5

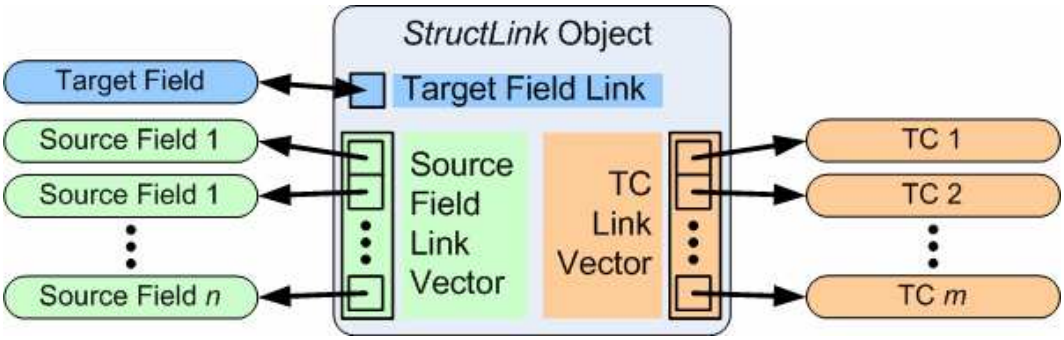

Figure 5.5: Example of a *StructLink* object.

Associations need to be defined for each of the required target fields, although this may present a problem if the source format does not contain all of the fields necessary to populate the fields in the target format. This issue is addressed in the TM implementation by allowing TCs to perform some operations, such as string manipulation, rather than referencing a source field. As a result, a required target field for which there was a corresponding source field could be populated with a null string, or a selected string such as "n/a", for example, based on the user's determination of what is appropriate.

A logical representation of the use of *StructLink* objects is shown in Figure 5.6, in which the associations defined in Figure 4.16 are represented.

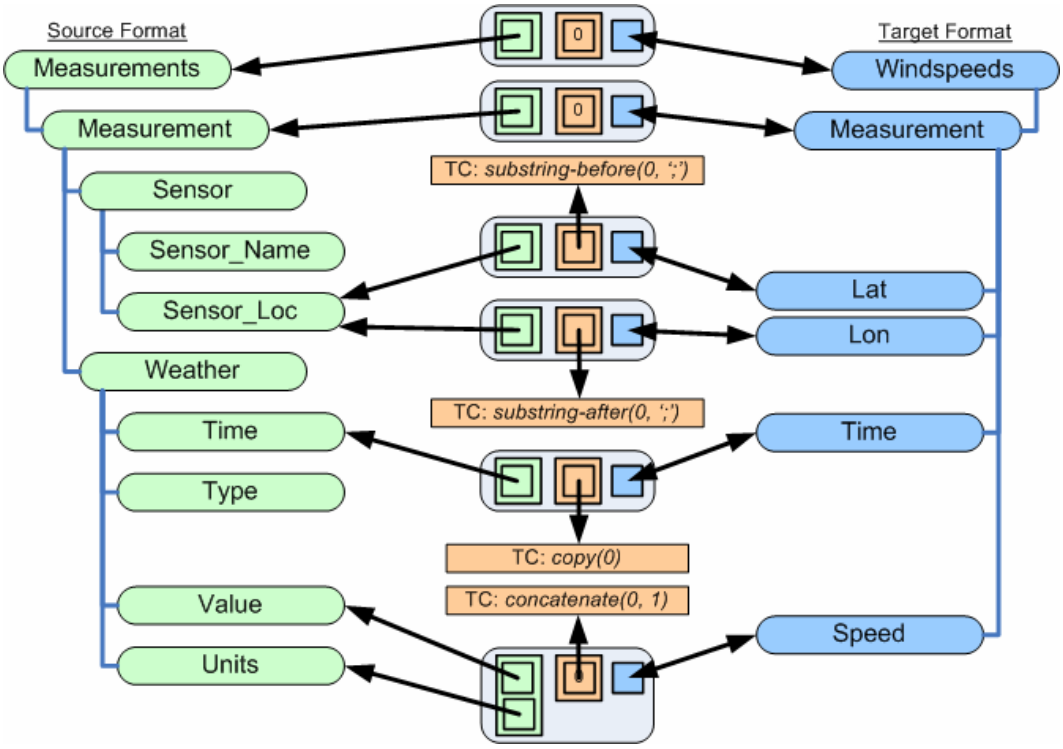

Figure 5.6: *StructLink* objects used to represent the associations defined in Figure 4.16.

Once all of the necessary associations have been defined, the TM traverses the target format structure (which is comprised of *StructNode* objects), and uses the data stored in *StructLink* object at each step to iteratively build the XSLT transformation document based on the associated source fields.

Using the example associations depicted in Figure 5.6, the following steps are performed by the TM to automatically generate the XSLT code:

1. The XSLT header is written as follows

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" />
```

2. The target field WINDSPEEDS is associated with the MEASUREMENTS source field, so the XML document root code is added to use the template for the MEASUREMENTS field.

```
<xsl:template match="/">
  <xsl:apply-templates select="MEASUREMENTS" />
</xsl:template>
```

3. The MEASUREMENTS template, which is the XSLT code used to generate the target field WINDSPEEDS from the source field MEASUREMENTS is added, which in this case involves the use of the MEASUREMENT template.

```
<xsl:template match="MEASUREMENTS" >
  <xsl:element name="WINDSPEEDS" >
    <xsl:apply-templates select="MEASUREMENT">
    </xsl:apply-templates>
  </xsl:element>
</xsl:template>
```

4. The MEASUREMENT template, which is the XSLT code used to generate the target field MEASUREMENT from the source field MEASUREMENT is added, which in this case involves the use of the MEASUREMENT template. The MEASUREMENT target field is composed of four child fields, and includes a selection criterion in which MEASURMENT data is only used if the TYPE child field has the value ‘windspeed’.

```
<xsl:template match="MEASUREMENT" >
  <xsl:if test="./WEATHER/TYPE='windspeed'">
    <xsl:element name="MEASUREMENT" >
      <xsl:call-template name="SENSOR_LOC_TO_LAT" >
        <xsl:with-param name="value1"
          select="SENSOR/SENSOR_LOC" />
      </xsl:call-template>
      <xsl:call-template name="SENSOR_LOC_TO_LON" >
        <xsl:with-param name="value1"
          select="SENSOR/SENSOR_LOC" />
      </xsl:call-template>
      <xsl:apply-templates select="WEATHER/TIME" />
      <xsl:apply-templates select="WEATHER/VALUE" />
```

```
    </xsl:element>
  </xsl:if>
</xsl:template>
```

5. The XSLT code used to generate the LAT target field is added, which utilizes an XSLT transformation component called *substring-before()*.

```
<xsl:template name="SENSOR_LOC_TO_LAT" >
  <xsl:param name="value1" />
  <xsl:element name="LAT" >
    <xsl:value-of select="substring-before($value1,':')" />
  </xsl:element>
</xsl:template>
```

6. The code to generate the LON field is added in a similar manner to the previous step, with the exception that *substring-after()* is used in place of *substring-before()*.

```
<xsl:template name="SENSOR_LOC_TO_LON" >
  <xsl:param name="value1" />
  <xsl:element name="LON" >
    <xsl:value-of select="substring-after($value1,':')" />
  </xsl:element>
</xsl:template>
```

7. The code to generate the TIME target field is added, which simply involves copying the value of the TIME field from the source to the target.

```
<xsl:template match="TIME" >
  <xsl:element name="TIME" >
    <xsl:value-of select="." />
  </xsl:element>
</xsl:template>
```

8. The last target field is SPEED, which is comprised from the concatenation of the VALUE and UNITS source fields. The context of this template is the VALUE field, so while the contents of the VALUE field can be easily acquired, the contents of other fields in the hierarchy (in this case UNITS) must be referenced in relation to the VALUE field. XSLT uses XPath [ClDe99, Ber05] expressions

to describe the relationship of fields to each other in a document hierarchy, and a function named *buildRelativePath()* in the *XsltGenerator* class determines the appropriate XPath expression to describe the relationship between two *StructNode* objects, which are passed to the function as parameters. In this case, the XPath from SPEED to UNITS is "../UNITS".

```
<xsl:template match="VALUE" >
  <xsl:element name="SPEED" >
    <xsl:value-of select="." />
    <xsl:value-of select="../UNITS" />
  </xsl:element>
</xsl:template>
```

9. A final line of XSLT code is added to complete the document.

```
</xsl:stylesheet>
```

The automatically generated XSLT document which describes the newly created transformation is shown in Figure 5.7. Once the XSLT document has been built, it is used to perform the current transformation. It is also saved for future use, and a record describing it is added to the TRANSFORMATIONS table in the TM database, ensuring that any effort required to generate a given transformation is not duplicated.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" />
  <xsl:template match="/">
    <xsl:apply-templates select="MEASUREMENTS" />
  </xsl:template>
  <xsl:template match="MEASUREMENTS" >
    <xsl:element name="WINDSPEEDS" >
      <xsl:apply-templates select="MEASUREMENT">
      </xsl:apply-templates>
    </xsl:element>
  </xsl:template>
  <xsl:template match="MEASUREMENT" >
    <xsl:if test="./WEATHER/TYPE='windspeed'">
      <xsl:element name="MEASUREMENT" >
        <xsl:call-template name="SENSOR_LOC_TO_LAT" >
          <xsl:with-param name="value1" select="SENSOR/SENSOR_LOC" />
        </xsl:call-template>
        <xsl:call-template name="SENSOR_LOC_TO_LON" >
          <xsl:with-param name="value1" select="SENSOR/SENSOR_LOC" />
        </xsl:call-template>
        <xsl:apply-templates select="WEATHER/TIME" />
        <xsl:apply-templates select="WEATHER/VALUE" />
      </xsl:element>
    </xsl:if>
  </xsl:template>
  <xsl:template name="SENSOR_LOC_TO_LAT" >
    <xsl:param name="value1" />
    <xsl:element name="LAT" >
      <xsl:value-of select="substring-before($value1,':')" />
    </xsl:element>
  </xsl:template>
  <xsl:template name="SENSOR_LOC_TO_LON" >
    <xsl:param name="value1" />
    <xsl:element name="LON" >
      <xsl:value-of select="substring-after($value1,':')" />
    </xsl:element>
  </xsl:template>
  <xsl:template match="TIME" >
    <xsl:element name="TIME" >
      <xsl:value-of select="." />
    </xsl:element>
  </xsl:template>
  <xsl:template match="VALUE" >
    <xsl:element name="SPEED" >
      <xsl:value-of select="." />
      <xsl:value-of select="../UNITS" />
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

Figure 5.7: Example of a completed XSLT document generated by the TM [NaHa05].

Actions

Requests to the TM Engine can include actions, which are invoked based on the outcome of a transformation request. There are six defined outcomes with which actions can be associated, as shown in Table 5.4.

Action Name	Description
ACT_SUCCESS	The transformation request completed successfully.
ACT_ERROR	An error was encountered during the transformation process, and as a result the transformation process was not completed.
ACT_TRANS_NOT_FOUND	A suitable transformation sequence was not found, and as a result the transformation process was not completed.
ACT_TRANS_FAILED	One of the transformations failed to complete successfully, and as a result the transformation process was not completed.
ACT_INPUT_FAILED	A failure was encountered while attempting to read the source data, and as a result the transformation process was not completed.
ACT_OUPUT_FAILED	A failure was encountered while attempting to write the target data, and as a result the transformation process was not completed.

Table 5.4: Defined TM outcomes with which actions can be associated.

The actions in the transformation request are stored in by the TM in a vector, so that a given outcome may result in one or more actions being invoked. Two actions types are defined in the TM, one of which allows an email to be sent, and the other which sends a message to the IDACT Scheduler [Lis03, Lis05, LNH04], which may in turn trigger further operations. The TM implementation was designed so that extending the number of actions is a simple task which involves adding a new class that extends the *ActionItem* class and its single *perform()* method, as shown below.

```
/**
 * Abstract base class for the various ActionItem* classes,
 * which are used to describe the details of an Action.
 */
public abstract class ActionItem extends Object
{
    abstract public boolean perform(String msg, DtmError err);
}
```

TM User Interface

The TM Engine is written so that it is not dependant on any single user interface (UI) design decision. Interaction between a UI and the engine is facilitated by a defined API which the UI can utilize to access the TM Engine functionality. As a result of the independence, a UI appropriate to the target user group can be used for any given TM instance.

Command Line Interface

A command line interface (CLI) was developed to test the TM. It provides full access to the TM functionality in a text-based format. While this UI may not be an appropriate choice for the typical end-user, it is extremely useful in a development environment, as it can easily be modified to incorporate debugging information in addition to the standard user information.

Graphical User Interface

As part of the IDACT project, a graphical user interface (GUI) is being developed that provides a consistent user interface across all of the IDACT components, including

the TM. This GUI development also involves a web-service layer to augment the current API, allowing TM functionality to be easily invoked by both local and remote user interfaces. The development of the GUI is beyond the scope of this work, although it is expected that such a UI will be more commonly used than the prototype test command line interface.

Security Issues

During the development process, some user feedback indicated the need to restrict access to data sources based on the identity of the user making the request. In response to this feedback, the IDACT Security Module (SM) was designed to allow for the implementation of such restrictions, without requiring major modification to each of the TM components which would need to be "security aware" [Wu04]. The relevant IDACT components (the Data Warehouse, Query Manager, Scheduler, and Transformation Manager) can now query the SM for permission to perform an action, such as retrieving data from a datasource, or even revealing the existence of a data source. The SM stores the relevant permissions for each user, and can determine which actions are permitted, and which are prohibited, on a per-user basis.

Source Code

The TM component of IDACT is available under the open source GNU Public License (GPL). The source code for the TM can be found in Appendix A.

TESTING

Overview

The Transaction Manager prototype has been tested in several environments using both black box and white box testing techniques. Unit tests were performed during the development of the TM as well as integration testing. This chapter focuses on the higher level integration testing, which initially included artificially generated test situations, formats, and datasets. The TM has also been tested using both realistic and real datasets to ensure that it is effective in domain-specific operating environments.

Test Environments

Testing Tools

The testing platform for the Transaction Manager was a system running the Java Runtime Environment version 1.4.2, on which several additional Java libraries had been installed, including the Xalan, Xerces, Java Activation Framework, and Java Mail libraries. In addition, transformations were installed on the system, some of which were created simply for testing purposes, and others which were real transformation tools. The required databases (DSR and TM) were also available on the test host, which was running a MySQL RDBMS server.

As the TM uses a command line interface, many of the tests were scripted using *expect* [Lib05], to allow for repeated entry of command line operations without requiring the presence of a human operator.

Transformation Sequence Testing

During development and debugging of the TM, several test scenarios were developed to determine the effectiveness of the TM in a range of circumstances. An example of the first such scenario involved populating the TRANSFORMATIONS table in the TM database with a series of test records that described transformations between several test formats, labeled A though E. In addition, several executable transformation simulators were written, corresponding to each of the test transformations. The transformations can be represented by the graph shown in Figure 6.1, and the relevant subset of the TRANSFORMATION table is shown in Table 6.1.

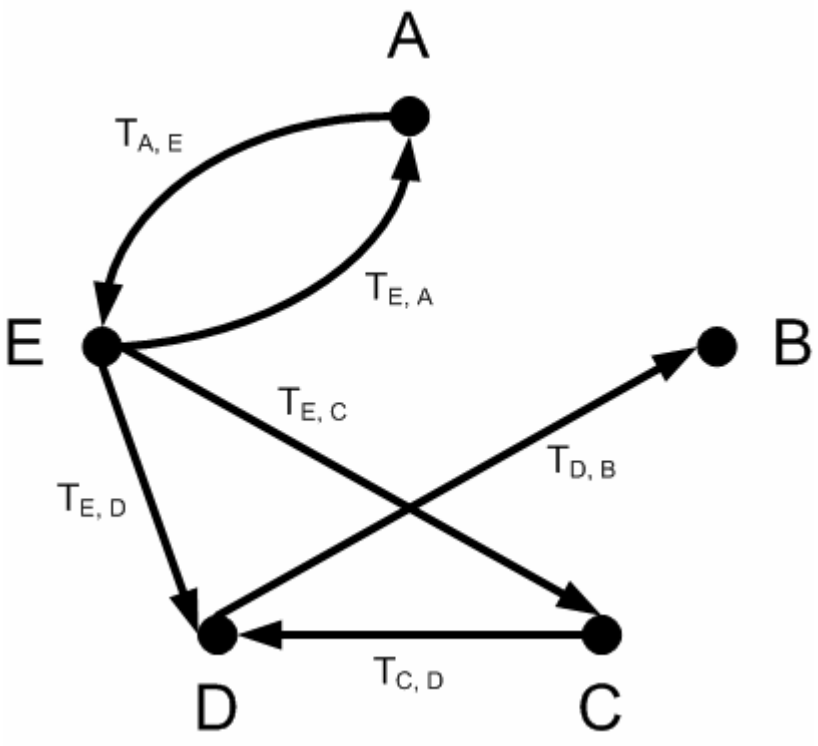

Figure 6.1: A graph representation of the six test transformations between five test formats.

ID	SRC_FORMAT	TGT_FORMAT	TRANSFORM_STR	Other Fields...
1	A	E	a2e.exe	...
2	E	A	e2a.exe	...
3	E	D	e2d.exe	...
4	E	C	e2c.exe	...
5	D	B	d2b.exe	...
6	C	D	c2d.exe	...

Table 6.1: The TRANSFORMATION table records corresponding to the transformations depicted in Figure 6.1.

The transformation simulators (e.g. *a2e.exe*) were initially configured to display the simulator name, and return 0, to indicate that the simulated transformation had completed successfully. An example of such a transformation simulator is shown in the following C code:

```
#include <stdio.h>

int main(int argc, char* argv[])
{
        printf("Performing transformation %s\n", argv[0]);
        return 0;
}
```

The first test in this example environment involved submitting a sequence of 25 transformation requests to the TM, with each request using one of the possible format pairs. For this test, no actions were specified, and the custom transformation code was stubbed out, to ensure that the test focused on the transformation sequence component only. The results of the test were compared with the expected results, as shown in Table 6.2, to ensure that they matched.

SOURCE FORMAT	TARGET FORMAT	EXPECTED TRANSFORMATION
A	A	Identity
A	B	A -> E -> D -> B
A	C	A -> E -> C
A	D	A -> E -> D
A	E	A -> E
B	A	No sequence found
B	B	Identity
B	C	No sequence found
B	D	No sequence found
B	E	No sequence found
C	A	No sequence found
C	B	C -> D -> B
C	C	Identity
C	D	C -> D
C	E	No sequence found
D	A	No sequence found
D	B	D -> B
D	C	No sequence found
D	D	Identity
D	E	No sequence found
E	A	E -> A
E	B	E -> D -> B
E	C	E -> C
E	D	E -> D
E	E	Identity

Table 6.2: The 25 possible transformation requests for the five defined test formats, and the associated expected transformation sequence.

A similar test was executed, with the exception that a non-existent format *F* was used as the source and/or target format, to ensure that the TM would respond appropriately in such cases (i.e. that a suitable transformation sequence was not found).

Action Testing

The next major component of the TM to be tested was the invocation of actions, both on the successful fulfillment of a transformation request, and in the event of some failure in the transformation process. Using the set of transformations described on page

79, a series of transformation requests were made to the TM. Different email actions were assigned to the ACT_SUCCESS, ACT_ERROR, and ACT_TRANS_NOT_FOUND events, which were the three possible outcomes for this test environment. The expected outcomes are shown in Table 6.3.

REQUEST	EXPECTED TRANSFORMATION SEQUENCE	EXPECTED ACTION EVENT
A -> A	Identity	ACT_SUCCESS
A -> B	A -> E -> D -> B	ACT_SUCCESS
A -> C	A -> E -> C	ACT_SUCCESS
A -> D	A -> E -> D	ACT_SUCCESS
A -> E	A -> E	ACT_SUCCESS
B -> A	No sequence found	ACT_TRANSFORM_NOT_FOUND
B -> B	Identity	ACT_SUCCESS
B -> C	No sequence found	ACT_TRANSFORM_NOT_FOUND
B -> D	No sequence found	ACT_TRANSFORM_NOT_FOUND
B -> E	No sequence found	ACT_TRANSFORM_NOT_FOUND
C -> A	No sequence found	ACT_TRANSFORM_NOT_FOUND
C -> B	C -> D -> B	ACT_SUCCESS
C -> C	Identity	ACT_SUCCESS
C -> D	C -> D	ACT_SUCCESS
C -> E	No sequence found	ACT_TRANSFORM_NOT_FOUND
D -> A	No sequence found	ACT_TRANSFORM_NOT_FOUND
D -> B	D -> B	ACT_SUCCESS
D -> C	No sequence found	ACT_TRANSFORM_NOT_FOUND
D -> D	Identity	ACT_SUCCESS
D -> E	No sequence found	ACT_TRANSFORM_NOT_FOUND
E -> A	E -> A	ACT_SUCCESS
E -> B	E -> D -> B	ACT_SUCCESS
E -> C	E -> C	ACT_SUCCESS
E -> D	E -> D	ACT_SUCCESS
E -> E	Identity	ACT_SUCCESS

Table 6.3: The test transformation requests and the corresponding expected action events for the first phase of action testing.

A second phase of action-focused testing was conducted, in which the executable transformation *e2d* was implemented so that it would return a failure code (-1), triggering the ACT_TRANS_FAILED event in these cases. This transformation (*e2d*) was selected for deletion because it appears in several transformation sequences, and it appears as the

first item, the middle item, or the last item depending on which transformation sequence is used. The tests performed, and the expected corresponding action events for this phase are shown in Table 6.4.

REQUEST	EXPECTED TRANSFORMATION SEQUENCE	EXPECTED ACTION EVENT
A -> A	Identity	ACT_SUCCESS
A -> B	A -> E -> D -> B	ACT_TRANS_FAILED
A -> C	A -> E -> C	ACT_SUCCESS
A -> D	A -> E -> D	ACT_TRANS_FAILED
A -> E	A -> E	ACT_SUCCESS
B -> A	No sequence found	ACT_TRANSFORM_NOT_FOUND
B -> B	Identity	ACT_SUCCESS
B -> C	No sequence found	ACT_TRANSFORM_NOT_FOUND
B -> D	No sequence found	ACT_TRANSFORM_NOT_FOUND
B -> E	No sequence found	ACT_TRANSFORM_NOT_FOUND
C -> A	No sequence found	ACT_TRANSFORM_NOT_FOUND
C -> B	C -> D -> B	ACT_SUCCESS
C -> C	Identity	ACT_SUCCESS
C -> D	C -> D	ACT_SUCCESS
C -> E	No sequence found	ACT_TRANSFORM_NOT_FOUND
D -> A	No sequence found	ACT_TRANSFORM_NOT_FOUND
D -> B	D -> B	ACT_SUCCESS
D -> C	No sequence found	ACT_TRANSFORM_NOT_FOUND
D -> D	Identity	ACT_SUCCESS
D -> E	No sequence found	ACT_TRANSFORM_NOT_FOUND
E -> A	E -> A	ACT_SUCCESS
E -> B	E -> D -> B	ACT_TRANS_FAILED
E -> C	E -> C	ACT_SUCCESS
E -> D	E -> D	ACT_TRANS_FAILED
E -> E	Identity	ACT_SUCCESS

Table 6.4: The test transformation requests and the corresponding expected action events for the second phase of action testing.

A third phase of action focused testing examined the operation of the TM when multiple actions were associated with a given action event. In this case, the executable transformation *e2d.exe* was again configured to return an error code, and the transformation requests shown in Table 6.5 were submitted to the TM.

REQUEST	EXPECTED TRANSFORMATION SEQUENCE	EXPECTED ACTION EVENT
A -> A	Identity	ACT_SUCCESS
A -> B	A -> E -> D -> B	ACT_TRANS_FAILED
A -> C	A -> E -> C	ACT_SUCCESS
A -> D	A -> E -> D	ACT_TRANS_FAILED
A -> E	A -> E	ACT_SUCCESS
A -> F	No sequence found	ACT_TRANSFORM_NOT_FOUND

Table 6.5: The test transformation requests and the corresponding expected action events for the third phase of action testing.

Action testing was also incorporated into the other test scenarios described in later sections of this Chapter, as each of those tests has a resulting expected outcome, which could be associated with a given action to easily and automatically determine the results of the test.

Transformation Testing

A subsequent series of tests involved the use of real transformations, and real datasets. Several transformations tools were identified [Bru05, Hay04, Jpg05], and others were manually developed for these tests. A set of transformation requests which utilized these transformations was created, and then submitted to the TM.

An example of such a scenario, which was identified by NASA for testing the IDACT system, involves the processing of telemetry data from a NASA/UAF sounding rocket, due for launch from Poker Flat Research Range in Spring 2007 (The initial launch date of February 2005 was postponed to ensure that adequate testing of the new on-board sensor hardware was performed). In similar sounding rockets missions in the past, archiving the telemetry data has been performed with little effort, but the analysis of the

data has been problematic. This presents an excellent demonstration and test environment for the TM, in which the data is transformed from the telemetry stream format into several data products. One such example transformation, which was used for this testing phase, was to generate graphical representations of various fields in the telemetry data, such as altitude. Although a program could be written to perform this transformation in a single step, it was possible to leverage existing tools and the TM transformation sequence process to more quickly create a method for transforming the data. In this case, the following existing tools were utilized:

1. An existing tool which parsed the telemetry stream format into fields, and stored the results as records in a database [Hay04].
2. An existing tool which creates an XML representation of a database table [Bos97, Vas02].
3. An existing tool which accepts an XML document as input, and generates a graphical representation of the data contained in the document [Jpg05].

Using these tools, it is almost possible to perform the required task without resorting to writing any new transformations. However, the XML format of the documents generated by step 2 does not match the XML format required for input to step 3. The TM addressed this problem, allowing a custom transformation to be defined which transformed data from the step 2 output format into the step 3 input format, and as a result a complete (four step) path from the telemetry stream source to the graphical representation was available to process the data.

The telemetry stream source format and graphical output target format was then used as one of the examples for this testing phase. All four transformation steps were defined in the TRANSFORMATIONS table of the TM database, and a request was submitted to transform a given telemetry stream dataset into a graphical representation, requiring the TM to determine then execute the relevant transformation sequence. This successful test was demonstrated for NASA representatives in April 2005.

Performance Testing

XSLT is a useful form in which to build custom transformations in that it is entirely platform independent, and can be easily modified or reused. However, it lacks the run-time efficiency of a compiled language, such as C, and this could be an issue for larger datasets. In an effort to determine how effective XSLT can be for larger datasets, a series of performance tests were developed and run.

Prior to testing, several XML datasets of various sizes were generated, corresponding to the schema shown in Figure 6.2. The XSLT transformation shown in Figure 6.3 was then used to process the datasets, using the Apache Xalan XSLT processor in both the Java and C++ versions [Xal05a, Xal05b]. The execution time for each transformation was recorded.

In order to address some of the performance issues associated with XSLT, modern XSLT processors such as Xalan provide the option to compile a given XSLT document into an alternate form, such as Java bytecode or C++, which should then perform the same operations more efficiently than the source XSLT document [Jor01,

Xal05a]. The compiled versions of XSLT documents are commonly referred to as *translets*. C++ and Java translets were created for the XSLT transformation shown in Figure 6.3, and these translets were then applied to the various XML source datasets, and the executing times were again recorded.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:complexType name="SENSOR_TYPE">
    <xsd:sequence>
      <xsd:element name="SENSOR_NAME" type="xsd:string" minOccurs="1" maxOccurs="1" />
      <xsd:element name="SENSOR_LOC" type="xsd:string" minOccurs="1" maxOccurs="1" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="MEASUREMENT_TYPE">
    <xsd:sequence>
      <xsd:element name="SENSOR" type="SENSOR_TYPE"
        minOccurs="1" maxOccurs="1" />
      <xsd:element name="WEATHER" type="WEATHER_TYPE"
        minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="WEATHER_TYPE">
    <xsd:sequence>
      <xsd:element name="TIME" type="xsd:dateTime" minOccurs="1" maxOccurs="1" />
      <xsd:element name="TYPE" type="xsd:string" minOccurs="1" maxOccurs="1" />
      <xsd:element name="VALUE" type="xsd:integer" minOccurs="1" maxOccurs="1" />
      <xsd:element name="UNITS" type="xsd:string" minOccurs="1" maxOccurs="1" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="MEASUREMENTS">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="MEASUREMENT" type="MEASUREMENT_TYPE"
          minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

</xsd:schema>
```

Figure 6.2: XML Schema description of example source format [NaHa05].

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" />
  <xsl:template match="/">
    <xsl:apply-templates select="MEASUREMENTS" />
  </xsl:template>
  <xsl:template match="MEASUREMENTS" >
    <xsl:element name="WINDSPEEDS" >
      <xsl:apply-templates select="MEASUREMENT">
      </xsl:apply-templates>
    </xsl:element>
  </xsl:template>
  <xsl:template match="MEASUREMENT" >
    <xsl:if test="./WEATHER/TYPE='windspeed'">
      <xsl:element name="MEASUREMENT" >
        <xsl:call-template name="SENSOR_LOC_TO_LAT" >
          <xsl:with-param name="value1" select="SENSOR/SENSOR_LOC" />
        </xsl:call-template>
        <xsl:call-template name="SENSOR_LOC_TO_LON" >
          <xsl:with-param name="value1" select="SENSOR/SENSOR_LOC" />
        </xsl:call-template>
        <xsl:apply-templates select="WEATHER/TIME" />
        <xsl:apply-templates select="WEATHER/VALUE" />
      </xsl:element>
    </xsl:if>
  </xsl:template>
  <xsl:template name="SENSOR_LOC_TO_LAT" >
    <xsl:param name="value1" />
    <xsl:element name="LAT" >
      <xsl:value-of select="substring-before($value1,':')" />
    </xsl:element>
  </xsl:template>
  <xsl:template name="SENSOR_LOC_TO_LON" >
    <xsl:param name="value1" />
    <xsl:element name="LON" >
      <xsl:value-of select="substring-after($value1,':')" />
    </xsl:element>
  </xsl:template>
  <xsl:template match="TIME" >
    <xsl:element name="TIME" >
      <xsl:value-of select="." />
    </xsl:element>
  </xsl:template>
  <xsl:template match="VALUE" >
    <xsl:element name="SPEED" >
      <xsl:value-of select="." />
      <xsl:value-of select="../UNITS" />
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

Figure 6.3: The XSLT transformation used to process the XML datasets[NaHa05].

Custom Transformation Generation Testing

In order to test the operation of the custom transformation generation component of the TM, several example XML Schemas were generated, and all records were removed from the TRANSFORMATIONS table in the TM database, and from the individual user domains in the DSR database. Several transformation requests involving these formats were submitted to the TM to determine how effectively associations between fields were defined, and to ensure that the defined associations were then applied in the expected manner in subsequent transformation requests. The custom transformation programs generated by the transformation manager were able to successfully transform the data from the input format to the required output format. The newly generated custom transformation programs were then added to the TRANSFORMATIONS table and additional transformation tests were conducted that utilized the new transformation. These included utilizing the new transformation in both singular transformations and transformation sequences. The TM was able to successfully transform the code in all scenarios.

Evaluation Methods

During the development cycle of the TM, there have been several points at which user evaluation has been conducted, and the resulting feedback used to guide the development process, ensuring that the result of this effort is not merely an intellectual exercise, but rather a product that has value for real data producers, archivists, and consumers.

The first evaluation group was NASA, which, as the primary funding agency, required periodic evaluations. These evaluations involved quarterly written progress reports, semi-annual teleconference reviews, annual face-to-face reviews and demonstrations, and two conference presentations. At each review, the project team is required to demonstrate that feedback from previous review cycles has been incorporated into the project. NASA funded the IDACT project because they, as an organization, face many of the problems that IDACT aims to address, namely that they have a wealth of data in geographically distributed, heterogeneous archives which can only be fully utilized if data consumers can actually access it in a suitable format. The NASA review process requires that the progress to date formally be accepted by NASA after the review. NASA formally accepted all IDACT reviews.

The second evaluation group involved personnel from the Aerospace Corporation [AC05] who described several real-world challenges they face as data consumed, which the TM could be used to address.

The third evaluation group involved a University of Alaska Fairbanks (UAF} team working on Sounding Rocket development, who utilized the TM to transform telemetry data from the native frame-based format used in the vehicle downlink to several formats for data archive and display, including a series of database INSERT statements, and a graphical view of several vehicle status values and instrument measurements.

One major issue which resulted from the feedback to date was the inability of the IDACT system, including the TM component, as initially designed to provide for access

on a per-user basis to a subset of the available datasets. This feedback motivated the work on the IDACT Security Module, as described on page 77.

Evaluation of IDACT as a whole, or of the TM component individually is a continuing process, and in addition to continuing work with the previously described groups, two additional avenues involving UAF scientists in the areas of biology and mining engineering are being pursued for the application and evaluation of the TM. These applications domains are both increasingly reliant on data management and manipulation, and the results of these collaborations will continue to drive the development of the TM.

CONCLUSIONS

Conclusions

This research effort addresses problems related to the integration of heterogeneous datasets, which are of great importance if the vast, diverse, and ever-expanding scientific, commercial, and governmental data resources are to be fully exploited. The underlying goal of this work is to reduce the level of technical skill required on the part of data producers and consumers, allowing them to focus on the use of the data, rather than the management details which are becoming increasingly time consuming. To that end, methods by which data transformation requests could be fulfilled by automatically identifying suitable sequences of existing transformations were provided. In addition, automated techniques for building new custom transformations based on previous experience and a library of transformation components were developed and described.

A prototype tool (the Transformation Manager) which utilizes these techniques was implemented and tested, and successfully demonstrated that these methods can be effective in addressing the issues of data transformation between heterogeneous data formats. Although the user remains in control of the process, automation is used to perform previously identified actions again should a similar or related circumstance arise. As a result, the knowledge of each TM user can be leveraged to solve problems from other users, rather than each user being required to solve every data processing problem individually, as is often the case today.

While all of the functions of the Transaction Manager advance the state of the art in intelligent automation, this research effort results in a significant contribution to the body of knowledge in computer science by providing a methodology that intelligently and automatically determines associations between heterogeneous data formats, and generates the computer code required to perform new transformations between such formats.

Future Work

While this effort extends the body of knowledge in computer science, and has practical applications for the data management domain, there are several avenues in which the results of this work could be extended to address other related issues.

The issue of what constitutes the "best" transformation sequence could be addressed in greater detail, with the goal of determining which metrics could be most usefully applied as weights for each transformation. While computational time required is an obvious choice for consideration, other metrics, such as transformation precision or computational space may also be applicable in some cases. In addition, it may not be possible to determine an accurate weight for computational effort, for example, until the actual dataset to be processed is examined, as the weight may be dependant not only on the size of the file, but on the relative size of elements within a particular file. A determination of which metrics are useful, and how they would be used in assigning a weight to each transformation is an excellent domain for future research efforts.

The issue of the transformation component language is also an area for further study. At this time the custom transformations are written using XSLT, based on XSLT

components. Increased performance may be possible if the transformation components were written as functions in a language such as C, which could then be compiled to executable code.

REFERENCES CITED

[AC05] Aerospace Corporation. (n.d.). Retrieved November 22, 2005 from http://www.aero.org/.

[Ada94] Adamek, J. (1994). Fusion: combining data from separate sources. *Marketing Research: A Magazine of Management and Applications*, 6, 48.

[AMA05] *Arctic Monitoring and Assessment Programme.* (n.d.). Retrieved August 14, 2003 from http://www.amap.no/

[AMA98] Arctic Monitoring and Assessment Programme. (1998). *AMAP Assessment Report: Arctic Pollution Issues.* Oslo, Norway: AMAP.

[Apa05] *The Apache XML Project*. (n.d.). Retrieved September 10, 2004, from http://xml.apache.org/

[ARM05] Atmospheric Radiation Monitoring Program. (n.d.). Retrieved November 22, 2005 from http://www.arm.gov/.

[Bay03] Baynton, P. (2003, June 18). Data integration or fusion. Which is best for mixed media schedule analysis? *ARF/ESOMAR Week of Audience Measurement*. Los Angeles, California, USA.

[Ber05] Berglund, A., Boag, S., Chamberlin, D., Fernandez, M., Kay, M., Robie, J., & Simeon, J. (Eds.). (n.d.). *XML Path Language (XPATH) Version 2.0*. Retrieved September 15, 2005 from the W3C Web site: http://www.w3.org/TR/2005/CR-xpath20-20051103/

[Bos97] *XML representation of a relational database*. (1997). Retrieved September 10, 2004, from http://www.w3.org/XML/RDB.html

[Bro02] Brown, J.G. (2002). Using a multiple imputation technique to merge data sets. *Applied Economic Letters*, 9, 311-314.

[Bru05] Bruss, S.P. (2005). IDACT interface and client software library and documentation. Retrieved March 3, 2005, from http://www.uaf.edu/asgp/spbruss/asgp/srp5_gse/srp5_gse.html

[CAD01] Castano, C., Antonellis, V.D., & diVimercati, S.D. (2001). Global Viewing of Heterogeneous Data Sources, *IEEE Transactions on Knowledge and Data Engineering*, v.13 n.2, p.277-297.

[CEH04] Cai, Z., Eisenhauer, G., He, Q., Kumar, V., Schwan, K., & Wolf, M. (2004). IQ-services: network-aware middleware for interactive large-data applications. *Proceedings of the 2nd Workshop on Middleware For Grid Computing (Toronto, Ontario, Canada, October 18 - 22, 2004).* ACM Press, New York, NY, 11-16.

[Cir05] *Stat/Transfer: the easiest way to move data between worksheets, databases, and statistical packages*. (n.d.). Retrieved from the Circle Systems Web site: http://www .stattransfer.com/

[ClDe99] Clark, J., & S. DeRose, (Eds.). (n.d.) XML Path Language (XPATH) Version 1.0. Retrieved September 15, 2004 from http://www.w3.org/TR/xpath

[Cov01] Cover, R. (Ed.). (n.d.). *Bioinformatic Sequence Markup Language (BSML).* Retrieved September 15, 2005 from http://xml.coverpages.org/bsml.html

[Cov05] Cover, R. (Ed.). (n.d.). *National Library of Medicine (NLM) XML Data Formats.* Retrieved September 15, 2005 from http://xml.coverpages.org/nlmXML.html

[Cre03] Crewdson, C. (2003). *Data Source Registry (DSR) Technical Specification*. University of Alaska Fairbanks Department of Mathematical Sciences. T-2003-435-1.

[Cre05] Crewdson, C. (2005). *IDACT: Data Source Registry*. M.S. Project Report. University of Alaska Fairbanks Department of Mathematical Sciences.

[CRS91] Clogg, C.C., Rubin, D.B., Schenker, N., Schultz, B. & Weidman, L. (1991) Multiple imputation of industry and occupation codes in census public-use samples using Bayesian logistic regression. *Journal of the American Statistical Association, 86*, 68-78.

[Das97] Dasarathy, B. V. (1997). Sensor fusion potential exploitation - Innovative architectures and illustrative applications. *Proceedings of the IEEE, 85*, pp. 24-38.

[Dat05] *Data Conversion and Migration Tools Survey*. (n.d.). Retrieved September 15, 2005 from http://dataconv.org/apps.html

[DLD04] Dhamankar, R., Lee, Y., Doan, A., Halevy, A., Domingos, P. (2004). iMAP: discovering complex semantic matches between database schemas. *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data. June 13-18, 2004*, Paris, France.

[DMD03] Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., & Halevy, A. (2003). Learning to match ontologies on the Semantic Web. *The International Journal on Very Large Data Bases, v.12 n.4.* 303-319.

[DMR03] Dimitoglou, G., Moore, P., & Rotenstreich, S. (2003). Middleware for large distributed systems and organizations. *Proceedings of the 1st International Symposium on Information and Communication Technologies (Dublin, Ireland, September 24 - 26, 2003). ACM International Conference Proceeding Series, vol. 49*. 536-542.

[DTC03] DeHaan, D., Toman, D., Consens, M. P., & Özsu, M. T. (2003). A comprehensive XQuery to SQL translation using dynamic interval encoding. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (San Diego, California, June 09 - 12, 2003). SIGMOD '03*. New York: ACM Press. 623-634.

[EDS05] *Environmental Data Standards.* (n.d.) Retrieved May 15, 2004 from http://www.envdatastandards.net/section/standards/?PHPSESSID=b0026e50cdde18df59714e92fa1b31ae

[EJX01] D. Embley, Jackman, D., & Xu, L. (2001). Multifaceted exploitation of metadata for attribute match discovery in information integration. *Proceedings of the International Workshop on Information Integration on the Web (WIIW'01), Rio de Janeiro, Brazil, April 2001.* 110-117.

[EXD04] Embley, D.W., Xu, L., & Ding, Y. (2004). Automatic direct and indirect schema mapping: experiences and lessons learned. *SIGMOD Rec. 33, 4*. 14-19.

[FGD05] *Geospatial Metadata Standards.* (n.d.) Retrieved May 15, 2004 from the Federal Geographic Data Committee Web site: http://www.fgdc.gov/metadata/meta_stand.html

[FoBo98] Foody, G.M. & Boyd, D.S. (1998). Mapping tropical forest biophysical properties from coarse spatial resolution satellite sensor data: applications of neural networks and data fusion. *Proceedings of the Third International Conference on GeoComputation, University of Bristol,17-19 September 1998.*

[FYL05] Fan, W., Yu, J.X., Lu, H., Lu, J., & Rastogi, R. (2005). Query translation from XPATH to SQL in the presence of recursive DTDs. *Proceedings of the 31st international Conference on Very Large Data Bases (Trondheim, Norway, August 30 - September 02, 2005)*. VLDB Endowment. 337-348.

[GDA05] GDA Implementation Statement. (n.d.). Retrieved November 22, 2005 from http://www.ga.gov.au/nmd/products/fwdprogram/ausliggda.jsp.

[GNS00] Gomes, J.I.C, Nagarajan, J., Shah, S. & Clement, C.L.T. (2000). An Experimental Infrastructure for the Sharing of Knowledge on the Internet: The iOn Project. *Proceedings of the 11th International Workshop on Database and Expert Systems Applications, September 2000.* 463.

[Hall97] Hall, D.L. & Llinas, J. (1997). An introduction to multisensor data fusion. *Proceedings of the IEEE, 85*, 6-23.

[HaNa00] Hay, B.& Nance, K. (2000). SIMON: An Intelligent Agent For Heterogeneous Data Mapping. *Proceedings of the International Conference on Intelligent Systems and Control. Honolulu, Hawaii. August 13-18, 2000.*

[HaNa04] Hay, B, & Nance, K. (2004). IDACT: Automating Scientific Knowledge Discovery. *Proceedings of the 7th IASTED Conference on Environmental Modelling and Simulation. November 2004.*

[Hay00] Hay, B. (2000). *Simon: An Intelligent Agent For Heterogeneous Data Compilation*. M.S. Project Report. University of Alaska Fairbanks Department of Mathematical Sciences.

[Hay03] Hay, B. (2003). *IDACT: Data Transformation Manager (DTM) Technical Specification.* University of Alaska Fairbanks Department of Mathematical Sciences. T-2003-434-1.

[Hay04] Hay, B. (2004). *Sounding Rocket Telemetry Stream Archiver.* University of Alaska Fairbanks Department of Mathematical Sciences. T-2004-227-1.

[ICE05a] *International Council for the Exploration of the Sea.* (n.d.) Retrieved September 3, 2005 from http://www.ices.dk/indexfla.asp

[ICE05a] *International Council for the Exploration of the Sea.* (n.d.) Retrieved November 24, 2005 from http://www.ices.dk/datacentre/reco/reco.asp

[ICO05] *ASCII to EBCDIC Conversions.* (n.d.) Retrieved September 3, 2005 from http://www.iconv.com/asciiebcdic.htm

[ITS05] Earth Science Markup Language, ESML. (n.d.). Retrieved from the Information Technology and Systems Center Web site: http://esml.itsc.uah.edu/index.jsp

[JMS02] Jain, S., Mahajan, R., and Suciu, D. (2002). Translating XSLT programs to Efficient SQL queries. *Proceedings of the 11th international Conference on World Wide Web (Honolulu, Hawaii, USA, May 07 - 11, 2002*. ACM Press, New York, NY, 616-626.

[Jor01] Jørgensen, M. (n.d.) *XSLTC Documentation.* Retrieved September 15, 2005 from the XSLTC Documentation Web site: http://xml.apache.org/xalan-j/xsltc/index.html

[Jpg04] *JpGraph – PHP Graph Creating Library*. (n.d.) Retrieved July 20, 2005 from http://www.aditus.nu/jpgraph/

[KJW02] Karlsson, B., Jarrhed, J.O., & Wide, P. (2002). A Fusion Toolbox for Sensor Data Fusion in Industrial Recycling. *IEEE Transactions on Instrumentation and Measurement, 51*, 1.

[KKN04] Krishnamurthy, R., Kaushik, R., & Naughton, J. F. (2004). Unraveling the duplicate-elimination problem in XML-to-SQL query translation. *Proceedings of the 7th international Workshop on the Web and Databases: Colocated with ACM SIGMOD/PODS 2004 (Paris, France, June 17 - 18, 2004).* New York: ACM Press, 49-54.

[Lib05] Libes, D. (n.d.). *The Expect Home Page*, Retrieved September 15, 2005 from the NIST Web site: http://expect.nist.gov/

[LiCl00] Li, W., & Clifton, C. (2000). SEMINT: a tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data & Knowledge Engineering, 33*, 1, 49-84.

[Lis03] Lisee, M. (2003). *IDACT: Scheduler Technical Specification.* University of Alaska Fairbanks Department of Mathematical Sciences. T-2003-436-1.

[Lis05] Lisee, M. (2005). *IDACT: Scheduler.* M.S. Project Report. University of Alaska Fairbanks Department of Mathematical Sciences. 2005.

[LMC02] Lee, D., Mani, M., Chiu, F., & Chu, W. W. (2002). NeT & CoT: translating relational schemas to XML schemas using semantic constraints. *Proceedings of the Eleventh international Conference on information and Knowledge Management (McLean, Virginia, USA, November 04 - 09, 2002).* New York: ACM Press. 282-291.

[LNH04] Lisee, M., Nance, K., & B. Hay. (2004). HTEST: A Configurable Triggered Data System for Simulating Space Systems Data Source Integration. *Proceedings of the 23rd Space Simulation Conference. November 8-11, 2004*. Washington, D.C.: NASA Goddard Space Flight Center.

[LYH02] Lee, M. L., Yang, L. H., Hsu, W., &Yang, X. (2002). XClust: clustering XML schemas for effective integration. *Proceedings of the Eleventh international Conference on information and Knowledge Management (McLean, Virginia, USA, November 04 - 09, 2002*., New York: ACM Press.

[MBR01] Madhavan, J., Bernstein, P.A., & Rahm, E. (2001). Generic Schema Matching with Cupid. *Proceedings of the 27th International Conference on Very Large Data Bases September 11-14, 2001*, 49-58.

[MMR02] Melnik, D., Molina-Garcia, U. & Rahm, E. (2002). Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching, *Proceedings of the 18th International Conference on Data Engineering (ICDE'02) February 26-March 01, 2002*. 117.

[MyS05] *MySQL* (n.d.). Retrieved September 15, 2005 from http://www.mysql.com/

[NaHa01] Nance, K, & Hay, B. (2001, February 7) *Presentation to the International Council for the Exploration of the Sea (ICES)*. Copenhagen, Denmark.

[NaHa04a] Nance, K. & Hay, B. (2004, May 4). Automating Conversions Between Metadata Standards Using XML and XSLT. *Presentation at the 2004 DAMA Symposium and Wilshire Metadata Conference, Beverly Hill, California, May 2-4, 2004.*

[NaHa04b] Nance, K. & Hay, B. (2004). IDACT: Automating Data Discovery and Compilation. *Proceedings of the 2004 NASA Earth Science Technology Conference, June 22-24 Palo Alto, CA.*

[NaHa05] Nance, K. & Hay, B. (2005). Automatic Transformations Between Geoscience Standards Using XML. *Computer & Geosciences Journal: Special Issue on XML Applications in Geosciences*. 31, 1165-1174.

[Nan97a] Nance, K. (1997a). Data System Planning for Formerly Used Defense Sites (FUDS). *Proceedings of the American Society of Business and Behavioral Sciences: Government and Business Problems. Las Vegas, NV. February 20-26, 1997.*

[Nan97b] Nance, K. (1997b). Decision Support and Data Mining. *Proceedings of the International Simulation Multiconference. April 6 – 10, 1997.*

[Nan97c] Nance, K. (1997c). Synthesis of Heterogeneous Data Sets. *Proceedings of the 9th Annual Software Technology Conference. Salt Lake City, UT, May 6 – 10, 1997.*

[Nan97d] Nance, K. (1997d). Applying AI Techniques in Developing Plans for Formerly Used Defense Sites (FUDS) in Alaska. *Mathematical Modeling and Scientific Computing, vol. 8.*

[NaSt00] Nance, K. & Strohmaier, M. (2000). The SynCon Project: Communicating Potential Health Effects of Contaminants to Circumpolar Arctic Communities. *Arctic Research of the United States, Special Issue on Arctic Health.*

[NaWi98] Nance, K. & Wiens, J.. (1998). SynCon: Simulating Remediation Planning. *Mathematical Modeling and Scientific Computing, vol. 9.*

[NCS05b] *Software Using HDF5 by Application Type.* (n.d.). Retrieved September 3, 2005 from the National Center for Supercomputing Applications Web site: http://hdf.ncsa.uiuc.edu/tools5app.html

[NWG99a] Nance, K, Wiens J., & George, S. (1999a). The SynCon Project: Arctic Regional Environmental Contamination Assessment. *Proceedings of the 38th Annual Western Regional Science Conference. Ojai, CA February 21-25, 1999.*

[NWG99b] Nance, K, Wiens J., & George, S. (1999b). The SynCon Project: Phase II Assessing Human Health in the Arctic. *Proceedings of the International ICSC Congress on Computational Intelligence: Methods and Applications. Rochester, NY. June 22-25, 1999.*

[NWS05] *National Digital Forecast Database (NDFD Extensible Markup Language (XML).* (n.d.). Retrieved from the National Weather Service Web site: http://www.weather.gov/xml/

[PHP05] *php*. (n.d.). Retrieved September 15, 2005 from http://www.php.net/

[PrGu03] Priebe, T. & Pernul, G. (2003). Towards integrative enterprise knowledge portals. *Proceedings of the twelfth international conference on Information and knowledge management table of contents. Pgs. 216 – 223. New Orleans, LA*.

[PVH02] Popa, L., Velegrakis, Y., Hernandez, M., Miller, R., & Fagin, R. (2002). Translating web data. *Proceedings of the 28th International Conference on Very Large Databases (VLDB'02), Hong Kong, China, August 2002*. 598-609.

[RaBe01] Rahm, E., & Bernstein, P.A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal — The International Journal on Very Large Data Bases, v.10 n.4,*334-350.

[RaNa98] Race, P., & Nance, K. (1998). Synthesizing and Assessing Arctic Contamination Data. *Proceedings of the American Society of Business and Behavioral Sciences: Information Systems. Las Vegas, NV. February 20-26, 1998*

[Rei01] Reierson, L.O. (2001). AMAP Secretariat. Letter dated October 31, 2001.

[Rus02] Rushbrook, T. (2002, June). Canadian Advertising Research Foundation Data Integration Committee Report. *Canadian Advertising Research Foundation Newsletter, 255.*

[SRB05] Storage Resource Broker. (n.d.). Retrieved November 22, 2005 from http://www.sdsc.edu/srb/.

[Sou05] *XSLT Standard Library*. (n.d.).Retrieved September 15, 2004 from the SourceForge.net Web site: http://sourceforge.net/projects/xsltsl/

[Sun05] *Java Technology*. (n.d.). Retrieved September 15, 2004 from the Sun Developer Network Web site: http://java.sun.com/

[TeNa98] Tedor, J. & Nance, K. Developing Decision Support Tools for Environmental Data Organization. *Proceedings of the American Society of Business and Behavioral Sciences: Decision Sciences. Las Vegas, NV. February 20-26, 1998.*

[ThSh04] Thomas, B. & Shaya, E. (n.d.). *eXtensible Data Format*. Retrieved September 15, 2005 from the eXtensible Data Format (XDF) Homepage Web site: http://xml.gsfc .nasa.gov/XDF/XDF_home.html

[Uni05] *NetCDF*. (n.d.). Retrieved September 3, 2004 from the Unidata Web site: http://www.unidata.ucar.edu/software/netcdf/

[US91] U.S. Department of the Interior and Minerals Management Service. (1991). *Outer Continental Shelf Environmental Assessment Program Final Reports of Principal Investigators. Vol. 23 (Oct. 1984) – vol. 74 (Oct 1991).*

[Vas02] Dynamically Generating XML from a Database. (n.d.). Retrieved September 10, 2004 from http://www.quepublishing.com/articles/article.asp?p=27800&rl=1

[W3C05a] *Extensible Markup Language (XML).* (n.d.). Retrieved May 15, 2004 from the W3C Working Group Web site: http://www.w3.org/XML/

[W3C05b] *Ink Markup Language (InkML).* (n.d.). Retrieved September 15, 2005 from the W3C Multimodal Interaction Working Group Web site: http://www.w3.org/2002/ mmi/ink

[W3C05c] *MathML.* (n.d.). Retrieved September 15, 2005 from the W3C Math Working Group Web site: http://www.w3.org/Math/

[WCH02] Wolf, M., Cai, Z., Huang, W., & Schwan, K. (2002). SmartPointers: personalized scientific data portals in your hand. *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing (Baltimore, Maryland). Conference on High Performance Networking and Computing.* IEEE Computer Society Press, Los Alamitos, CA, 1-16.

[WSW05] Wiseman, Y., Schwan, K., & Widener, P. (2005). Efficient end to end data exchange using configurable compression. *SIGOPS Oper. Syst. Rev. 39, 3* 4-23.

[Wu04] Wu, C. (2004). *IDACT: Security System Specification.* University of Alaska Fairbanks Department of Mathematical Sciences. T-2004-245-1.

[Xal05a] *Xalan-C++ version 1.8.* (n.d.). Retrieved May 15, 2004 from the Apache XML Project Web site: http://xml.apache.org/xalan-c/

[Xal05b] *Xalan-Java version 2.7.0.* (n.d.). Retrieved May 15, 2004 from the Apache XML Project Web site: http://xml.apache.org/xalan-j/

[YuPo02] Yu, C. & Popa, L. (2004). Constraint-based XML query rewriting for data integration. *Proceedings of the 2004 ACM SIGMOD international Conference on Management of Data (Paris, France, June 13 - 18, 2004).* ACM Press, New York, NY, 371-382.

APPENDIX A

Source Code

```
/**
 * File:            Dtm.java
 *
 * @author:         Brian Hay
 *
 * @version:        $Id: Dtm.java,v 1.5 2005/04/23 23:30:44 bhay Exp $
 *
 */

/* package definition */
package Dtm;


/* imports - IDACT specific */
import IdactHelpers.*;
import IdactHelpers.Actions.*;
import Dtm.Transformers.*;
import Dtm.Errors.*;
import Dtm.Config.*;


/* imports - general */
import java.sql.*;
import java.net.*;
import java.util.Calendar;


/**
 * A class that provides Level 1 (control) functionality for the transformation
manager.  This is the
 * high level controller that runs all of the transformation manager
functionality.
 */
public class Dtm
{

   /* private fields */
   private DtmTransformer dtmt[] = {    // array of available transformation
types
       new DtmTransformerXslt(),
       null,
       new DtmTransformerExec()
   };
   private DtmError err = new DtmError();  // error object

   // TM database connection fields
   final String dbHost = "localhost";
   final String dbUser = "idact";
   final String dbPass = "JX?^@Jh7CM";
   final String dbName = "TM";

   /**
    * constructor
    */
   public Dtm()
   {
       // set up error object for success
       err.set(DtmError.ERR_NONE, Action.ACT_SUCCESS, "", null);
   }
```

```
   /**
    * Process data based on a {@link IdactHelpers.ReqStruct}.  Attempts to
    * determine the transformations
    * necessary to get from the source data format to the desired data format,
and then
    * attempts to apply those transformations to the source data.
    *
    * @param rqs    an ReqStruct containing information about the requested
data transformation,
    *               such as the URI, type (stream or static) and
format.subformat
    *               (e.g. XML/messages.dtd) of the incoming and outgoing data.
It also
    *               contains the various Action object to be used in various
circumstances,
    *               such as a successful transformation, or an error.
    *
    * @return boolean indicating the success (true) or failure (false) of the
method.
    */
   public boolean processData(ReqStruct rqs)
   {
      // read the config file
      try
      {
         DtmConfig.initialize();
      }
      catch (Exception e)
      {
         System.err.println("DEBUG\tprocessData(): DtmConfig initialization
failed.");
         e.printStackTrace();
      }

      DataTransform[] dt = getTransformations(rqs, err);

      if (null != dt)
      {
         int transId = dbTransInProgress(rqs);

         applyTransformations(rqs, dt, err);

         performAction(rqs, err);

         dbTransCompleted(transId, err.getAction() );

         return(Action.ACT_SUCCESS == err.getAction() );
      }
      else
      {
         performAction(rqs, err);
         return false;
      }
   }


   /** Determine which action should be performed, and fire the Actions's
perform()
      method to actually do the work.
```

```
      Attempts to retrieve the action of the given action type (actType) in the
ReqStruct,
      and then fires the ActionHandler.perform() method.  If no matching action
type is found,
      the generic error action is used.

      Returns void.
   */
   private void performAction(ReqStruct rqs, DtmError err)
   {
      if (DtmConfig.DEBUG > 0) System.err.println("DEBUG\tperformAction():
actType=" + err.getAction() +
                                          "\n\tmsg='" + err.getMsg() + "'");

      Action a = rqs.getAction( err.getAction() );
      if ( null == a )
      {
         a = rqs.getAction(Action.ACT_ERROR);
      }
      if ( null != a )
      {
         a.perform( err.getMsg(), err );
      }
   }


   /**
    * Determime which transformation(s) should be applied, based on the input
    * and output formats.
    *
    * If a sequence of transformations can be found, an array of DataTransform
    * objects describing the sequence is returned.
    *
    * @param rqs
    * @param err
    *
    * @return
    */
   private DataTransform[] getTransformations(ReqStruct rqs, DtmError err)
   {
      Connection dbConn = null;  // database connection object
      DataTransform[] dt = null; // array of DataTransform objects to return
      Statement stmt;            // database statement object
      ResultSet rs;          // database result set object
      String query;          // database query string
      boolean seqFound = false;  // flag to indicate that sequence has been
found

      try
      {
         /*// connect to the database using the ODBC System DSN
         Driver d =
(Driver)Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
         String URL = "jdbc:odbc:" + "IDACT_DTM";
         dbConn = DriverManager.getConnection(URL, "", "");*/
         // connect to the database using the MySQL driver
         Driver d =
(Driver)Class.forName("com.mysql.jdbc.Driver").newInstance();
         dbConn = DriverManager.getConnection( ("jdbc:mysql://" + dbHost + "/"
+ dbName +
```

```
                                          "?user=" + dbUser + "&password=" +
dbPass) );
          if (DtmConfig.DEBUG > 0)
          {
             System.err.println("DEBUG\tgetTransformations(): db
Connected...");
          }

          int seqLen = 1;
          while ( (seqLen <= DtmConfig.TRANS_SEQ_MAX_LENGTH) && (!seqFound) )
          {
             //  build the query
             query = buildGetTransSql(seqLen, rqs);
             if (DtmConfig.DEBUG > 0)
             {
                System.err.println("DEBUG\tquery='" + query + "'");
             }

             // execute
             stmt = dbConn.createStatement();
             rs = stmt.executeQuery(query);

             // create and populate the dt array
             if (rs.next())
             {
                // create the transformation sequence array
                dt = new DataTransform[seqLen];
                int argGrp[] = new int[seqLen];

                // populate the transformation array
                for (int ii = 0; ii < seqLen; ii++)
                {
                   dt[ii] = new
DataTransform(DataTransform.strToClassification(rs.getString("TRANSFORM_TYPE_"
+ ii)),
                                                new
String(rs.getString("TRANSFORM_STRING_" + ii)));
                   // get the arguments for this transformation
                   argGrp[ii] = rs.getInt("TRANSFORM_ARG_GRP_" + ii);
                }
                for (int ii = 0; ii < seqLen; ii++)
                {
                   String argQuery = "SELECT `TA`.`ARG` FROM
`TRANSFORMATION_ARG_GRPS` TG, `TRANSFORMATION_ARGS` TA WHERE
`TG`.`ARG_ID`=`TA`.`ID` AND `TG`.`GRP_ID`=" + argGrp[ii] + " ORDER BY
TG.`ARG_ORDER`";
                   if (DtmConfig.DEBUG > 0)
                   {
                      System.err.println("DEBUG\targQuery='" + argQuery +
"'");
                   }
                   ResultSet argRs = stmt.executeQuery(argQuery);

                   if (null != argRs)
                   {
                      while (argRs.next())
                      {
                         dt[ii].addArg(argRs.getString("ARG"));
                      }
                   }
```

```
                }
                // set the transformation found flag
                seqFound = true;
            }
            else
            {
                seqLen++;
            }
        }

        // check for no transformation sequence found
        if ( !seqFound )
        {
            err.set(DtmError.ERR_NO_TRANS_FOUND, Action.ACT_TRANS_NOT_FOUND,
                   "No suitable transformation found.", null);
            dt = null;
        }

        // close the database connection
        dbConn.close();
        if (DtmConfig.DEBUG > 0)
System.err.println("DEBUG\tgetTransformations(): db connection closed.");
      }
      catch (Exception e)
      {
         // handle exceptions by setting error object, and setting dt to null
         err.set(DtmError.ERR_GET_TRANS, Action.ACT_TRANS_NOT_FOUND,
                "Error in Dtm.getTransformation(rqs): ", e);
         dt = null;
      }

      // return the DataTransform array, or null in case of error
      return  dt;
   }



   /** Determine which transformation type to apply, and call the
transformation's
      process() method to start the transformation.

      Parameters are a ReqStruct and an array of DataTransform objects, which
are applied
      in order.

      On success, true is returned.  On failure, false is returned and the
error object
      is updated to indicate the specific error.
   */
   private boolean applyTransformations(ReqStruct rqs, DataTransform dt[],
DtmError err)
   {
      // create a temp rqs to hold intermediate transformation information for
transformation sequences
      ReqStruct tempRqs = new ReqStruct();
      tempRqs.setDataIn( rqs.getDataIn() );


      // for each transformation we have
      for (int ii = 0; ii < dt.length; ii++)
```

```
      {
         // set the data output name
         if (ii < dt.length - 1)
         {
            // we need a temp name for the output
            tempRqs.setDataOut( "file://?f=" + DtmConfig.DATA_TEMP_DIR +
rqs.getIdNumber() + "-" + ii );
         }
         else
         {
            // use the final name for the output
            tempRqs.setDataOut( rqs.getDataOut() );
         }

         // set the arguments for the transformation
         tempRqs.setArgs();

         // if this is a valid transformation type
         if ( (dt[ii].getClassification() >= 0) && (dt[ii].getClassification()
< dtmt.length) )
         {
            // call the particular transformer to do the work
            if ( !( dtmt[dt[ii].getClassification()].process(tempRqs, dt[ii],
err) ) )
            {
               // transformation failed, so return false
               return false;
            }
         }
         else
         {
            // the transaction type was invalid, so set the error code and
return
            err.set( DtmError.ERR_INVAL_TRANS_TYPE,
Action.ACT_TRANS_NOT_FOUND,
                     "Transformation type '" + dt[ii].getClassification() + "'
in transformation '" + ii + "' is not valid.",
                     null);
            return false;
         }

         // move the temp data output name to the temp data input name for the
next iteration
         tempRqs.setDataIn( tempRqs.getDataOut() );
      }

      return true;
   }



   /**
    * This method build the SQL statement that is used to search for a
    * suitable transformation sequence.
    *
    * @param seqLen An integer that represents the length of the sequence that
the SQL
    *               statement should search for.  This value must be >= 1.
    * @param rqs
    *
```

```
    * @return A String containing the SQL statement that should be used to
search for
    *         a sutiable transformation sequence.
    */
   private String buildGetTransSql(int seqLen, ReqStruct rqs)
   {
      // start the query
      String sqlQuery = "SELECT TRANSFORMATIONS_0.TRANSFORM_TYPE AS
TRANSFORM_TYPE_0,TRANSFORMATIONS_0.ID AS TRANSFORM_ID_0,
TRANSFORMATIONS_0.TRANSFORM_NAME AS TRANSFORM_NAME_0,
TRANSFORMATIONS_0.TRANSFORM_STRING AS TRANSFORM_STRING_0,
TRANSFORMATIONS_0.TRANSFORM_ARG_GRP AS TRANSFORM_ARG_GRP_0  ";

      // add each other set of fields
      for (int ii = 1; ii < seqLen; ii++)
      {
         sqlQuery += ",TRANSFORMATIONS_" + ii + ".TRANSFORM_TYPE AS
TRANSFORM_TYPE_" + ii +
                     ", TRANSFORMATIONS_" + ii + ".ID AS TRANSFORM_ID_" + ii +
                     ", TRANSFORMATIONS_" + ii + ".TRANSFORM_NAME AS
TRANSFORM_NAME_" + ii +
                     ", TRANSFORMATIONS_" + ii + ".TRANSFORM_STRING AS
TRANSFORM_STRING_" + ii +
                     ", TRANSFORMATIONS_" + ii + ".TRANSFORM_TYPE AS
TRANSFORM_TYPE_" + ii +
                     ", TRANSFORMATIONS_" + ii + ".TRANSFORM_ARG_GRP AS
TRANSFORM_ARG_GRP_" + ii + " ";
      }

      // add the first part of the FROM CLAUSE
      sqlQuery += "FROM ";
      for (int ii = 1; ii < seqLen - 1; ii++)
      {
         sqlQuery += "( ";
      }
      sqlQuery += "TRANSFORMATIONS AS TRANSFORMATIONS_0 ";

      // add each JOIN clause
      for (int ii = 1; ii < seqLen; ii++)
      {
         sqlQuery += "INNER JOIN TRANSFORMATIONS AS TRANSFORMATIONS_" + ii + "
ON TRANSFORMATIONS_" + (ii - 1) + ".OUT_FMT = TRANSFORMATIONS_" + ii + ".IN_FMT
" +
                     "AND TRANSFORMATIONS_" + (ii - 1) + ".OUT_SUBFMT =
TRANSFORMATIONS_" + ii + ".IN_SUBFMT ";
         if (ii < seqLen - 1)
         {
            sqlQuery += ") ";
         }
      }

      // add the WHERE clause
      sqlQuery += "WHERE  ( (TRANSFORMATIONS_0.IN_FMT='" + rqs.getInFormat() +
"') AND (TRANSFORMATIONS_" + (seqLen - 1) + ".OUT_FMT='" + rqs.getOutFormat() +
"') " +
                  "AND (TRANSFORMATIONS_0.IN_SUBFMT='" + rqs.getInSubFormat() +
"') AND (TRANSFORMATIONS_" + (seqLen - 1) + ".OUT_SUBFMT='" +
rqs.getOutSubFormat() + "') )";

      if (DtmConfig.DEBUG > 0)
```

```
      {
         System.err.println("DEBUG\tAttempting to find transformation sequence
on len " + seqLen);
         System.err.println("\tquery='" + sqlQuery + "'");
      }

      return sqlQuery;

   }


   /**
    * This method adds a message to the database to indicate that a
    * transformation has started.
    *
    * @param rqs    A RequestStruct object containing the details of the
transformation.
    *
    * @return An int containing the record ID of the entry added to the
database
    *         for this transformation process.
    */
   private int dbTransInProgress(ReqStruct rqs)
   {
      // build the SQL query
      String sqlQuery = "INSERT INTO TRANSFORMATION_STATUS (INPUT, OUTPUT,
START_TIME, IN_PROGRESS, REQUEST_ID) VALUES ('" +
                        rqs.getDataIn() + "', '" + rqs.getDataOut() + "', '";
      Calendar startTime = Calendar.getInstance();
      String startTimeStr = startTime.get(Calendar.YEAR) + "-" +
                           startTime.get(Calendar.MONTH) + "-" +
                           startTime.get(Calendar.DAY_OF_MONTH) + " " +
                           startTime.get(Calendar.HOUR_OF_DAY) + ":" +
                           startTime.get(Calendar.MINUTE) + ":" +
                           startTime.get(Calendar.SECOND);
      sqlQuery += startTimeStr + "', TRUE, " + rqs.getIdNumber() + ")";

      // execute the query to add
      Connection dbConn = null;     // database connection object
      Statement stmt;               // database statement object
      ResultSet rs;              // database result set object
      try
      {
         Driver d =
(Driver)Class.forName("com.mysql.jdbc.Driver").newInstance();
         dbConn = DriverManager.getConnection( ("jdbc:mysql://" + dbHost + "/"
+ dbName +
                                               "?user=" + dbUser + "&password=" +
dbPass) );
         if (DtmConfig.DEBUG > 0)
         {
            System.err.println("DEBUG\tdbTransInProgress(): db Connected.");
         }

         // execute
         stmt = dbConn.createStatement();
         stmt.execute(sqlQuery);
      }
      catch (Exception e)
      {
```

```
            // handle exception
            if (DtmConfig.DEBUG > 0)
            {
               System.err.println("DEBUG\tdbTransInProgress(): Failed while
trying to add record.\n" + e.getMessage());
               System.err.println("DEBUG\t" + sqlQuery);
            }
            return -1;
         }

         // grab the ID number to return
         sqlQuery = "SELECT ID FROM TRANSFORMATION_STATUS WHERE INPUT='" +
rqs.getDataIn() +
                    "' AND OUTPUT='" + rqs.getDataOut() + "' AND REQUEST_ID=" +
rqs.getIdNumber() +
                    " AND START_TIME=#" + startTimeStr + "#";
         */
         sqlQuery = "SELECT REQUEST_ID FROM TRANSFORMATION_STATUS WHERE INPUT='" +
rqs.getDataIn() +
                    "' AND OUTPUT='" + rqs.getDataOut() + "' AND REQUEST_ID=" +
rqs.getIdNumber() +
                    " AND START_TIME='" + startTimeStr + "'";
         if (DtmConfig.DEBUG > 0)
         {
            System.err.println("DEBUG\tAttempting to get REQUEST_ID.\n\tquery='"
+ sqlQuery + "'");
         }

         try
         {
            // execute and return ID value
            rs = stmt.executeQuery(sqlQuery);
            if (rs.next())
            {
               return rs.getInt("REQUEST_ID");
            }
            dbConn.close();
            if (DtmConfig.DEBUG > 0)
            {
               System.err.println("DEBUG\tdbTransInProgress(): db closed.");
            }

         }
         catch (Exception e)
         {
            // handle exception
            if (DtmConfig.DEBUG > 0)
            {
               System.err.println("DEBUG\tdbTransInProgress(): Failed while
trying to find ID.\n" + e.getMessage());
               System.err.println("DEBUG\t" + sqlQuery);
            }
            return -1;
         }
         return -1;
      }



      /**
```

```
    * This method adds a message to the database to indicate that a
    * transformation has completed, including the completion status (e.g
success, failed,
    * etc).
    *
    * @param transId The record ID that relates to this transformation.
    * @param result  An int indicating the completion status, which should be
one of the
    *                ACTION.ACT_* flags.
    */
   private boolean dbTransCompleted(int transId, int completionStatus)
   {
      Connection dbConn = null;     // database connection object
      Statement stmt;               // database statement object
      ResultSet rs;              // database result set object

      // build the query
      Calendar endTime = Calendar.getInstance();
      String endTimeStr = endTime.get(Calendar.YEAR) + "-" +
                      endTime.get(Calendar.MONTH) + "-" +
                      endTime.get(Calendar.DAY_OF_MONTH) + " " +
                      endTime.get(Calendar.HOUR_OF_DAY) + ":" +
                      endTime.get(Calendar.MINUTE) + ":" +
                      endTime.get(Calendar.SECOND);
      String sqlQuery = "UPDATE TRANSFORMATION_STATUS SET IN_PROGRESS=FALSE,
COMPLETION_STATUS='" +
                       Action.getActionStr(completionStatus) + "', END_TIME='" +
endTimeStr +
                       "' WHERE REQUEST_ID=" + transId;

      // execute the query to update the record
      try
      {
         Driver d =
(Driver)Class.forName("com.mysql.jdbc.Driver").newInstance();
         dbConn = DriverManager.getConnection( ("jdbc:mysql://" + dbHost + "/"
+ dbName +
                                               "?user=" + dbUser + "&password=" +
dbPass) );
         if (DtmConfig.DEBUG > 0)
         {
            System.err.println("DEBUG\tdbTransCompleted(): DB Connected...");
            System.err.println("DEBUG\tquery='" + sqlQuery + "'");
         }

         // execute
         stmt = dbConn.createStatement();
         stmt.execute(sqlQuery);
         dbConn.close();
         if (DtmConfig.DEBUG > 0)
         {
            System.err.println("DEBUG\tdbTransCompleted(): DB Connection
Closed.");
         }
         return true;
      }
      catch (Exception e)
      {
         // handle exception
         if (DtmConfig.DEBUG > 0)
```

```
          {
             System.err.println("DEBUG\tdbTransCompleted(): Failed while trying
to update record.\n" + e.getMessage());
          }
          return false;
       }
   }
}
```

```
/**
 * File:            DtmConfig.java
 *
 * @author:         Brian Hay
 *
 * @version:        $Id: DtmConfig.java,v 1.4 2005/04/23 23:30:44 bhay Exp $
 *
 */

/* package definition */
package Dtm.Config;

/* imports - general */
import javax.mail.*;
import javax.mail.internet.*;

public class DtmConfig
{
    /* static debug constant to indicate debug level */
    public static int DEBUG;

    public static String SMTP_HOST;
    public static String SMTP_USER;
    public static String SMTP_PASS;

    public static Address EMAIL_FROM;

    public static int TRANS_SEQ_MAX_LENGTH;

    public static String DATA_TEMP_DIR;

    public static String EXEC_PATH;

    /**
     * constructor
     */
    public static void initialize() throws Exception
    {
        DEBUG = 0;

        SMTP_HOST = "mail.uaf.edu";
        SMTP_USER = "";
        SMTP_PASS = "";

        EMAIL_FROM = new InternetAddress("fnbh1@uaf.edu", "Brian Hay");

        TRANS_SEQ_MAX_LENGTH = 3;

        DATA_TEMP_DIR = "/idact/data/tmp/";

        EXEC_PATH = "/idact/transformations/bin/";
    }
}
```

```
/**
 * File:            DtmError.java
 *
 * @author:         Brian Hay
 *
 * @version:        $Id: DtmError.java,v 1.3 2004/06/10 06:43:13 bhay Exp $
 *
 */


/* package definition */
package Dtm.Errors;


/* imports - IDACT specific */
import Dtm.Config.*;
import IdactHelpers.Actions.*;

public class DtmError
{

    /* constants to define the various errors we have */
    public final static int ERR_NONE                = 0;        // No error

    public final static int ERR_INVAL_TRANS_TYPE    = 1001;     // Invalid
transformation type
    public final static int ERR_GET_TRANS           = 1002;     // Exception
raised when retreiving transformation from DB
    public final static int ERR_NO_TRANS_FOUND      = 1003;     // No suitable
transformation was found in the DTM database

    public final static int ERR_TRAN_XSLT_TCE       = 10001;    //
Transformation configuration exception in the XSLT Transformer
    public final static int ERR_TRAN_XSLT_TE        = 10002;    //
Transformation exception in the XSLT Transformer
    public final static int ERR_TRAN_XSLT_URIS      = 10003;    // URI syntax
exception in the XSLT Transformer

    public final static int ERR_TRAN_EXEC_IOE       = 20001;    // IO exception
in the EXEC Transformer
    public final static int ERR_TRAN_EXEC_IE        = 20002;    // Interrupted
exception in the EXEC Transformer
    public final static int ERR_TRAN_EXEC_RETVAL    = 20003;   // executable
returned unexpected value

    public final static int ERR_ACT_MAIL_ME_RECIP   = 100001;   // Failed to
set message recipient
    public final static int ERR_ACT_MAIL_ME_CONT    = 100002;   // Failed to
set message content
    public final static int ERR_ACT_MAIL_AE         = 100003;   // Failed t
    public final static int ERR_ACT_MAIL_ME_SUBJ    = 100004;   // Failed to
set message subject
    public final static int ERR_ACT_MAIL_ME_SNDR    = 100005;   // Failed to
set message sender.
    public final static int ERR_ACT_MAIL_ME_SEND    = 100006;   // Failed to
send message.

    /* private fields */
    private int errNum = ERR_NONE;                  // error number, defined as
static constants in this class
```

```
    private int errAction = Action.ACT_SUCCESS;     // action to be performed,
as defined by constants in ReqStruct class
    private String errMsg = "";                     // an error message


    /**
     * A method that sets values for the error number, error action,
     * and error messages.
     *
     * @param eNum   An int representing the error number.  Error numbers are
defined as static constants in this class.
     *
     * @param eAct   An int containing the error action (if known).  The
actions are defined as constants in
     *               {@link Action} class
     *
     * @param eMsg   A String containing the error message.
     *
     * @param e      An Exception object for the last error, which may be null.
     */
    public void set(int eNum, int eAct, String eMsg, Exception e)
    {
        // set the private field values based on the params.
        errNum = eNum;
        errAction = eAct;
        // check that the message object is not null
        if (null != eMsg)
        {
            // set the error message
            errMsg = new String( eMsg.trim() );
        }
        else
        {
            // use an empty string for the error message
            errMsg = new String("");
        }

        // if the debug flag is set, then print a message.
        if (DtmConfig.DEBUG > 0)
        {
            System.err.println("DEBUG\tError Number: " + eNum + "\n\t" +
eMsg);
            if (null != e)
            {
                System.err.println("Stack Trace:\n------------");
                e.printStackTrace();
            }
        }
    }


    /**
     * Get method for the error number.
     *
     * @return An int representing an error number, which are defined as
constants in this class.
     */
    public int getNum()
    {
        return errNum;
```

```
    }


    /**
     * Get method for the error action.
     *
     * @return An int representing an error action, which are defined as
constants in the
     *         {@link Action} class.
     */
    public int getAction()
    {
        return errAction;
    }


    /**
     * Get method for the error message.
     *
     * @return A String representing an error message.
     */
    public String getMsg()
    {
        return ( new String(errMsg) );
    }

}
```

```
/**
 * File:            DtmTransformer.java
 *
 * @author:         Brian Hay
 *
 * @version:        $Id: DtmTransformer.java,v 1.2 2004/06/06 04:18:10 bhay Exp
$
 *
 */


/* package definition */
package Dtm.Transformers;


/* imports - IDACT specific */
import IdactHelpers.*;
import Dtm.Errors.*;


/**
 * Abstract base class for the various Transformer* classes, which are used to
transform
 * one data format into another.
 */
public abstract class DtmTransformer extends Object
{
    abstract public boolean process(ReqStruct rqs, DataTransform dt, DtmError
err);
}
```

```
/**
 * File:            DtmTransformerExec.java
 *
 * @author:         Brian Hay
 *
 * @version:        $Id: DtmTransformerExec.java,v 1.5 2005/04/23 23:30:44 bhay
Exp $
 *
 */

/* package definition */
package Dtm.Transformers;

/* imports - IDACT specific */
import IdactHelpers.*;
import IdactHelpers.Actions.*;
import Dtm.Errors.*;
import Dtm.Config.*;

/* imports - general */
import java.io.*;
import java.util.Vector;

/**
 * A class that transforms data using an exectuable.
 */
public class DtmTransformerExec extends DtmTransformer
{
   /**
    * A method that attempts to apply a data transformation
    *
    * @param rqs    The {@link ReqStruct} object for this DTM run.
    * @param dt     A {@link DataTransform} object that descibes the data
transformation to
    *               apply.  In this case, it provides the command line
construction for the
    *               executable file.  The command line specification also
includes locations
    *               where command line parameters should be substituted, such
as input and
    *               output file names.
    * @param err    A {@link DtmError} object, which can be updated by this
method to
    *               provide information about any error which may occur during
the data
    *               transformation process.
    *
    * @return A boolean indicating whether or not the data transformation was
successful.
    */
   public boolean process(ReqStruct rqs, DataTransform dt, DtmError err)
   {
      Vector clVector = new Vector();
      clVector.add(DtmConfig.EXEC_PATH + dt.getTransformStr());
      for (int ii = 0; ii < dt.numArgs(); ii++)
      {
         String arg = dt.getArg(ii);
         int pcPos = arg.indexOf("%");
         while (pcPos >= 0)
         {
```

```
            // get the name of this argument
            String argName;
            int argLen = arg.substring(pcPos + 1).indexOf(" ");
            if (argLen < 0)
            {
               argName = arg.substring(pcPos + 1);
            }
            else
            {
               argName = arg.substring(pcPos + 1, pcPos + 1 + argLen);
            }
            if (DtmConfig.DEBUG > 0) { System.err.println("DEBUG\t" +
argName); }
            arg = arg.replaceAll("%" + argName, rqs.getArg("_" + argName));
            pcPos = arg.indexOf("%");
         }
         clVector.add(arg);
      }
      String[] clArray = new String[clVector.size()];
      if (DtmConfig.DEBUG > 0) { System.err.println("DEBUG\tclArray
contents:"); }
      for (int ii = 0; ii < clVector.size(); ii++)
      {
         clArray[ii] = (String)clVector.elementAt(ii);
         if (DtmConfig.DEBUG > 0) { System.err.println("\t" + ii + "'" +
clArray[ii] + "'"); }
      }

      // execute the command
      Process p;
      try
      {
         p = Runtime.getRuntime().exec( clArray );
      } catch (IOException ioe)
      {
         err.set( DtmError.ERR_TRAN_EXEC_IOE, Action.ACT_TRANS_FAILED,
                "EXEC command line IO exception", ioe);
         return false;
      }

      // set up and start the IO Threads for the command
      InputStream error = p.getErrorStream();
      InputStream output = p.getInputStream();
      Thread errThread = new Thread(new OutErrReader(error));
      Thread outThread = new Thread(new OutErrReader(output));
      outThread.start();
      errThread.start();

      // wait for the command return value
      int retVal;
      try
      {
         retVal = p.waitFor();
      } catch (InterruptedException ie)
      {
         err.set( DtmError.ERR_TRAN_EXEC_IE, Action.ACT_TRANS_FAILED,
                "EXEC Interrupted while waiting for command to complete", ie);
         return false;
      }
      // check return value
```

```
      if (retVal != 0)
      {
         err.set(DtmError.ERR_TRAN_EXEC_RETVAL, Action.ACT_TRANS_FAILED,
               "Unexpected return value from executable transformation(" +
retVal + ")", null);
         return false;
      }

      // all OK
      err.set(DtmError.ERR_NONE, Action.ACT_SUCCESS, "", null);
      return true;
   }

   /**
    * Private inner class to handle output and error streams from the
    * executable.
    *
    *
    * @author Brian Hay, based on code by CEHJ - from http://www.experts-
exchange.com/Programming/Programming_Languages/Java/Q_20966148.html
    */
   private class OutErrReader implements Runnable
   {
      InputStream is;

      /**
       * Constructor for the OutErrReader object
       *
       * @param is     An InputStream object, which will either be the output
or error streams
       *               for the executable.
       */
      public OutErrReader(InputStream is) {
         // set the input stream for this thread.
         this.is = is;
      }

      /**
       *  Main processing method for the OutErrReader object
       */
      public void run() {
         try
         {
            BufferedReader in = new BufferedReader(new InputStreamReader(is));
            String temp = null;

            // As long as there is data to be read from the stream
            while ((temp = in.readLine()) != null)
            {
               System.out.println(temp);
            }
            is.close();
         } catch (Exception e)
         {
            e.printStackTrace();
         }
      }
   }
}
```

```
/**
 * File:            DtmTransformerXslt.java
 *
 * @author:         Brian Hay
 *
 * @version:        $Id: DtmTransformerXslt.java,v 1.4 2005/04/23 23:30:44 bhay
Exp $
 *
 */

/* package definition */
package Dtm.Transformers;

/* imports - IDACT specific */
import IdactHelpers.*;
import IdactHelpers.Actions.*;
import Dtm.Errors.*;

/* imports - general */
import java.net.*;
import java.io.File;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;

/**
 * A class that transforms data using an XSLT stylesheet.  The input data
format is
 * usually XML, and the output data format can be almost anything.
 */
public class DtmTransformerXslt extends DtmTransformer
{
   private final String XSLT_PATH = "/idact/transformations/xslt";


   /**
     * A method that attempts to apply a data transformation
     *
     * @param rqs    The {@link ReqStruct} object for this DTM run.
     *
     * @param dt     A {@link DataTransform} object that descibes the data
transformation to
     *               apply.  In this case, it provides a URI for the XSLT
stylesheet to be used.
     *
     * @param err    A {@link DtmError} object, which can be updated by this
method to
     *               provide information about any error which may occur during
the data
     *               transformation process.
     *
     * @return A boolean indicating whether or not the data transformation was
successful.
     */
    public boolean process(ReqStruct rqs, DataTransform dt, DtmError err)
    {
        // attempt to set up and apply the transformation
```

```
        try
        {
            // set up the transformer
            TransformerFactory tf = TransformerFactory.newInstance();
            Transformer t = tf.newTransformer(new StreamSource( XSLT_PATH + "/"
+ dt.getTransformStr() ) );

            // perform the transformation
            // t.transform(new StreamSource(new File(new
URI(rqs.getDataIn()))), new StreamResult(new File(new URI(rqs.getDataOut()))));
          System.out.println("'" + rqs.getArg("_if") + "'");
          System.out.println("'" + rqs.getArg("_of") + "'");
            t.transform(new StreamSource(new File(new URI("file://" +
rqs.getArg("_if")))), new StreamResult(new File(new URI("file://" +
rqs.getArg("_of")))));
       }
        // handle exceptions - these all update the DtmError object and return
false immediately
        catch (TransformerConfigurationException tce)
        {
            err.set( DtmError.ERR_TRAN_XSLT_TCE, Action.ACT_TRANS_FAILED,
                    "XSLT transformer configuration failed", tce);
            return false;
        }
        catch (TransformerException te)
        {
            err.set( DtmError.ERR_TRAN_XSLT_TE, Action.ACT_TRANS_FAILED,
                    "XSLT transformation failed.", te);
            return false;
        }
        catch (URISyntaxException ue)
        {
            err.set( DtmError.ERR_TRAN_XSLT_URIS, Action.ACT_TRANS_FAILED,
                    "XSLT URI sytnax invalid.", ue);
            return false;
        }

        // if we got to here, then the transformation was successful.  Update
the error
        // object to reflect this, and return true.
        err.set(DtmError.ERR_NONE, Action.ACT_SUCCESS, "", null);
        return true;

    }
}
```

```
/**
 * File:            DataFormat.java
 *
 * @author:         Brian Hay
 *
 * @version:        $Id: DataFormat.java,v 1.2 2004/06/06 04:18:26 bhay Exp $
 *
 */


/* package definition */
package IdactHelpers;


/**
 * A class that describes a data format, which is used for both the input
 * data and the output data.
 * <P>
 * For example, the data may be XML (the format) which corresponds to a
 * particular DTD (the subformat).
 */
public class DataFormat
{
    /* private fields */
    private String format;          // data format, such as XML or HDF
    private String subFormat;       // data sub-format, such as a specific DTD


    /**
     * Default constructor - just initializes the object with empty strings.
Essentially this just
     *                       calls setFormat("", "")
     */
    public DataFormat()
    {
        // just use the other constructor with empty strings to do the work
here.
        this("", "");

    }   // DataFormat()


    /**
     * Constructor - just initializes the object using the parameters.
Essentially this just
     *               calls setFormat(f, subf)
     *
     * @param f      a String that contains the data format, such as "XML" or
"HDF"
     *
     * @param subf   a String that contains the data sub-format, such as a
specific DTD
     */
    public DataFormat(String f, String subf)
    {
        // just use setFormat(String, String) to do the work here
        setFormat(f, subf);

    }   // DataFormat()
```

```
    /**
     * Get method for data format.
     *
     * @return A String that contains the data format, such as "XML" or "HDF"
     */
    public String getFormat()
    {
        // return a copy of the format string
        return (new String(format));

    }   // getFormat()


    /**
     * Get method for data sub-format.
     *
     * @return A String that contains the data sub-format, such as a specific
DTD
     */
    public String getSubFormat()
    {
        // return a copy of the sub-format string
        return (new String(subFormat));

    }   // getSuFormat()


    /**
     * Set method for the data format.  Essentially just calls setFormat(f, "")
     *
     * @param f      A String containing the data format, such as "XML" or
"HDF"
     */
    public void setFormat(String f)
    {
        // just use setFormat(String, String) to do the work here
        setFormat(f, "");

    }   // setFormat()


    /**
     * Set method for the data format and sub-format.  This method is used
     * by many of the other methods in this class.
     *
     * Basically just sets values for the data format and sub-formats, but
     * it performs some validity checking on the parameters first.  For
example,
     * leading and trailing white space is removed, If the parameter f is
     * the empty string or null, then both the format and sub-format are set
     * to the empty string.  If the parameter sub-f is null, the sub-format
     * is set to the empty string.
     *
     * @param f      A String containin the data format, such as "XML" or "HDF"
     *
     * @param subf   A String containin the data sub-format, such as a specific
DTD
     */
    public void setFormat(String f, String subf)
    {
```

```
        // we must at least have a format value
        if ( (f != null) && ("" != (f = f.trim() ) ) )
        {
            format = new String(f);
            if (subf != null)
            {
                subFormat = new String(subf.trim() );
            }
            else
            {
                subFormat = new String("");
            }
        }
        // otherwise just use empty strings
        else
        {
            format = new String("");
            subFormat = new String("");
        }

    }   // setFormat()

}   // class DataFormat
```

```
/**
 * File:            DataTransform.java
 *
 * @author:         Brian Hay
 *
 * @version:        $Id: DataTransform.java,v 1.3 2005/04/23 23:30:44 bhay Exp
$
 *
 */


/* package definition */
package IdactHelpers;


/* imports - general */
import java.net.*;
import java.util.Vector;


/**
 * A class that describes a data transformation.
 * Essentially it stores a transformation classification, and
 * and a URI that contains specific instructions for the
 * transformation.
 * <P>
 * For example, a transformation may have an XSLT classification,
 * and the URI may be a specific XSLT stylesheet.
 */
public class DataTransform
{
    /* constants to define the various transformation classifications we have
*/
    public static final int CLASS_UNDEF     = -1;   // classification is
undefined
    public static final int CLASS_XSLT      =  0;   // classification is XSLT
    public static final int CLASS_BASH      =  1;   // classification is bash
shell
    public static final int CLASS_EXEC      =  2;   // classification is
executable
    public static final int CLASS_WEB_SERV  =  3;   // classification is web
service
   public static final int CLASS_PHP    =  4;  // classification is PHP
    private static final int CLASS_MAXVAL   =  4;   // maximum classification
value

    /* private fields */
    private int classification = CLASS_UNDEF;       // transformation
classification
    private String transformString;                 // transformation specific
string (command line, XSLT URI, etc).
   private Vector argVector;                   // vector of arguments to the
transformation

    /**
     * constructor - just initializes the object using the parameters.  If the
classification value
     *               is invalid, then CLASS_UNDEF is used.
     *
     * @param transformClass
```

```
     *               an int representing the transformation classification,
chosen from the
     *               constants defined in this class.
     * @param transformStr
     *               a String containing transformation specific data, such as
the URI for an XSLT file
     */
    public DataTransform(int transformClass, String transformStr)
    {
        // set the transformation class, which should correspond to one of the
CLASS_* constants
        // defined in this class.
        if ( (transformClass >= 0) && (transformClass <= CLASS_MAXVAL) )
        {
            classification = transformClass;
        }
        else
        {
            classification = CLASS_UNDEF;
        }

        // set the URI for the a specific transformation resource.
        transformString = new String( transformStr );

       argVector = new Vector();

    }   // DataTransform()

    /**
     * Get method for classification.
     *
     * @return An int representing the transformation classification.  This
value
     *         should be chosen from one of the CLASS_* constants defined in
this class
     */
    public int getClassification()
    {
        return classification;

    }   // getClassification()

    /**
     * Get method for the transformation specific string.
     *
     * @return An String for the resource providing transformation specific
details.
     */
    public String getTransformStr()
    {
        return ( new String( transformString ) );

    }   // getTransformStr()

   public void addArg(String a)
   {
      argVector.add(a);
   }

   public String getArg(int i)
```

```
    {
       return (String)argVector.elementAt(i);
    }

    public int numArgs()
    {
       return argVector.size();
    }


     /**
      * A method used to convert a String to a int that represent a
      * transformation classification.
      *
      * @param s      A String representing a transformation classification.
      *
      * @return An int representing the transformation classification, or
      *         CLASS_UNDEF if the parameter does not represent a valid
      *         transformation classification
      */
     public static int strToClassification(String s)
     {
         if (s.toUpperCase().equals("XSLT"))
         {
             return CLASS_XSLT;
         }
         else if (s.toUpperCase().equals("BASH"))
         {
             return CLASS_BASH;
         }
         else if (s.toUpperCase().equals("EXEC"))
         {
             return CLASS_EXEC;
         }
         else if (s.toUpperCase().equals("WSERV"))
         {
             return CLASS_WEB_SERV;
         }
         else if (s.toUpperCase().equals("PHP"))
         {
             return CLASS_EXEC;
         }
         else
         {
             return CLASS_UNDEF;
         }

     }   // strToClassification()

}   // class DataFormat
```

```
/**
 * File:            ReqStruct.java
 *
 * @author:         Brian Hay
 *
 * @version:        $Id: ReqStruct.java,v 1.5 2005/04/23 23:30:44 bhay Exp $
 *
 */


/* package definition */
package IdactHelpers;


/* imports - IDACT specific */
import IdactHelpers.Actions.*;
import Dtm.Config.*;


/* imports - general */
import java.net.*;
import java.util.Random;
import java.util.Hashtable;


/**
 * A class for containing the request structure in the Data Transformation
Manager (DTM).
 * <P>
 * Information related to the input and output data (such as URI, data
 * type, and data format) are stored in this object, as well as the
 * list of actions for each of the defined outcomes (including
 * success, general errors, and some specific error cases).
 */
public class ReqStruct
{

    /* private fields */
    private int idNumber;           // ID number for this request
    private String dataIn;          // name for the incoming data (e.g.
filename, URI, etc)
    private String dataOut;         // name for the outgoing data (e.g.
filename, URI, etc)
    private DataRate dataRateIn;    // data rate for incoming data (stream or
static)
    private DataRate dataRateOut;   // data rate for outgoing data
    private DataFormat dataFmtIn;   // data format for the incoming data (e.g.
XML/DTD, HDF, etc)
    private DataFormat dataFmtOut;  // data format for the outgoing data
    private Action[] actions;       // action array
   private Hashtable transformArgs;// hash table of arguments to the
transformation

    /**
     * Default constructor - essentially sets everything to null or -1
     */
    public ReqStruct()
    {
        idNumber = -1;
        dataIn = dataOut = null;
```

```
        dataRateIn = dataRateOut = null;
        dataFmtIn = dataFmtOut = null;
        actions = new Action[6];
        actions[0] = new Action("SUCCESS");
        actions[1] = new Action("ERROR");
        actions[2] = new Action("TRANSFORMATION NOT FOUND");
       actions[3] = new Action("TRANSFORMATION FAILED");
       actions[4] = actions[5] = null;

       transformArgs = new Hashtable();
    }   // ReqStruct()


    /**
     * Constructor that populates the object with meaningful values based on
the parameters
     *
     * @param in     A String of the input data
     *
     * @param out    A String for the output data
     *
     * @param dfIn   A {@link DataFormat} object describing the input data
format
     *
     * @param dfOut  A DataFormat object describing the utput data format
     *
     */
    public ReqStruct(String in, String out, DataFormat dfIn, DataFormat dfOut)
    {
        // call the default to do most of the work
        this();

       idNumber = Math.abs( new Random().nextInt() );

        dataIn = new String(in);
        dataOut = new String(out);

       // populate the hash table with the arguments
       populateHashTable();

        dataFmtIn = new DataFormat(dfIn.getFormat(), dfIn.getSubFormat());
        dataFmtOut = new DataFormat(dfOut.getFormat(), dfOut.getSubFormat());

        actions = new Action[6];
        actions[0] = new Action("SUCCESS");
        actions[1] = new Action("ERROR");
        actions[2] = actions[3] = actions[4] = actions[5] = null;

    }   // ReqStruct()


   // populate the hash table with the arguments
   private void populateHashTable()
   {
      String argStr = new String(dataIn);
      String io = "_i";

      for (int ii = 0; ii < 2; ii++)
      {
         // find the ? that starts the args
```

```
            int pos = argStr.indexOf("?");
            argStr = argStr.substring(pos + 1);

            boolean done = false;

            while (argStr.length() > 0)
            {
                // get the next argument
                int aLen = argStr.indexOf("&");
                if (aLen == -1)
                {
                    aLen = argStr.length();
                    done = true;
                }
                String nextArg = argStr.substring(0, aLen);
                int eqPos = nextArg.indexOf("=");
                String argName = nextArg.substring(0, eqPos);
                String argVal = nextArg.substring(eqPos + 1);

                // add the argument to the hashtable
                if (DtmConfig.DEBUG > 0)
                {
                    System.out.println("Found argument named '" + io + argName + "'
with value '" + argVal + "'");
                }
                transformArgs.put((io + argName), argVal);
                if (DtmConfig.DEBUG > 0)
                {
                    System.out.println("Argument named '" + io + argName + "' with
value '" +
                                       (String)transformArgs.get((io + argName)) +
"'");
                }

                // reset the string size
                if (done)
                {
                    argStr = "";
                }
                else
                {
                    argStr = argStr.substring(aLen + 1);
                }
            }

            io = "_o";
            argStr = new String(dataOut);
        }
    }


    /**
     * Get method for ID number
     *
     * @return An int containing the ID number
     */
    public int getIdNumber()
    {
        return idNumber;
```

```
    }   // getIdNumber()


   // return one of the transformation arguments
   public String getArg(String argName)
   {
      //System.out.println("getArg(" + argName + "), " + transformArgs.size());
      return (String)transformArgs.get(argName);
   }

   // set the arguments based on the dataIn and dataOut values
   public void setArgs()
   {
      transformArgs.clear();
      populateHashTable();
   }


    /**
     * Get method for input data String
     *
     * @return A String containing the source data (e.g. URI, filename, etc)
     */
    public String getDataIn()
    {
        return( new String(dataIn) );

    }   // getDataIn()


    /**
     * Set method for input data String
     *
     * @param newDataIn The String to be used for the data input (URI,
filename, etc).
     */
    public void setDataIn(String newDataIn)
    {
        dataIn = new String( newDataIn );

    }   // setDataIn()


    /**
     * Get method for output data String
     *
     * @return A String containing the data destination (e.g. URI, filename,
etc)
     */
    public String getDataOut()
    {
        return( new String(dataOut) );

    }   // getDataOut()


    /**
     * Set method for output data String
     *
     * @param newDataOut The String to be used for the data output (URI,
```

```
filename, etc).
     */
    public void setDataOut(String newDataOut)
    {
        dataOut = new String( newDataOut );

    }   // setDataOut()


    /**
     * Get method for input data format (e.g. XML)
     *
     * @return A String containing the input data format
     */
    public String getInFormat()
    {
        return dataFmtIn.getFormat();

    }   // getInFormat()


    /**
     * Get method for input data sub-format (e.g. messages.dtd)
     *
     * @return A String containing the input data sub-format
     */
    public String getInSubFormat()
    {
        return dataFmtIn.getSubFormat();

    }   // getInSubFormat()


    /**
     * Get method for output data format
     *
     * @return A String containing the output data format
     */
    public String getOutFormat()
    {
        return dataFmtOut.getFormat();

    }   // getOutFormat()


    /**
     * Get method for output data sub-format (e.g. messages.dtd)
     *
     * @return A String containing the output data sub-format
     */
    public String getOutSubFormat()
    {
        return dataFmtOut.getSubFormat();

    }   // getOutSubFormat()


    /**
     * Get method for an {@link Action}
     *
```

```
     * @param actType   An int indicating the action type.  This should be one
of the ACT_* constants defined
     *                  in the ReqStruct class.
     *
     * @return          An Action object if one exists, or null if there is no
suitable Action object defined
     */
    public Action getAction(int actType)
    {
        // return the requested action if it exists, or null otherwise
        if ( (actType >= 0) && (actType < actions.length ) )
        {
            return actions[actType];
        }
        return null;

    }   // getAction()


    /**
     * Add a new {@link ActionItem} to one of the defined actions.  An
ActionItem is an object
     * that descibes an event, such as send an email, that should occur as part
of an action.
     *
     * @param actType An int indicating the action type.  This should be one of
the ACT_* constants defined
     *                in the ReqStruct class.
     *
     * @param ai      An ActionItem object describing the action item to be
added.
     *
     * @return A boolean indicating whether the operation was successful or
not.
     */
    public boolean addActionItem(int actType, ActionItem ai)
    {
        // range check the actType value
        if ( (actType < 0) || (actType >= actions.length) )
        {
            // out of array range
            return false;
        }

        // make sure there is an action instanstiated
        if (actions[actType] == null)
        {
            actions[actType] = new Action();
        }

        // add the action item to the action
        return (actions[actType].addActionItem(ai));

    }   // addActionItem()

}   // class ReqStruct
```

```
/**
 * File:            Action.java
 *
 * @author:         Brian Hay
 *
 * @version:        $Id: Action.java,v 1.3 2005/04/17 18:18:59 bhay Exp $
 *
 */


/* package definition */
package IdactHelpers.Actions;

/* imports - IDACT specific */
import Dtm.Errors.*;
import Dtm.Config.*;

/* imports - general */
import java.util.*;


/**
 * A class to describe an action to be attempted for a particular
 * event.  Includes a public method perform() that causes the action
 * to be performed.
 */
public class Action
{
    /* constants to define the various actions we have */
    public final static int ACT_SUCCESS         = 0;    // No errors found
    public final static int ACT_ERROR           = 1;    // Generic error
    public final static int ACT_TRANS_NOT_FOUND = 2;    // Transformation-not-
found error
    public final static int ACT_TRANS_FAILED    = 3;    // Transformation-
failed error
    public final static int ACT_INPUT_FAILED    = 4;    // Input-data-access
failed
    public final static int ACT_OUPUT_FAILED    = 5;    // Output-data-access
failed
    private final static int ACT_MAXVAL         = 5;    // maximum valid ACT_*
value


    /* private fields */
    String actionName;                  // the name of this particular action
    Vector actItems = new Vector();     // vector of the ActionItems for this
action


    /**
     * Default constructor - just initializes the object with an empty string
     *                       as its name.  Essentially calls Action("") to
     *                       accomplish this.
     */
    public Action()
    {
        // call the Action(String) to do the work
        this("");

    }   // Action()
```

```
    /**
     * Constructor - just initializes the object using the parameter name for
     *               the action name.  It uses an empty string if the parameter
     *               is null, and trims leading and trailing white space from
the
     *               parameter in other cases.
     */
    public Action(String name)
    {
        // check for a non-null parameter
        if (null != name)
        {
            // set the actionName value to the trimmed parameter value
            actionName = new String( name.trim() );
        }
        // parameter is null, so use empty string
        else
        {
            actionName = new String("");
        }
    }   // Action()


    /**
     * This method converts an int value into a String corresponding to the
     * action type.
     *
     * @param a      The int value for the action, which should be one of the
Action.ACT_*
     *               values.
     *
     * @return A String corresponding to the action passed as a paremeter.
     */
    public static String getActionStr(int a)
    {
       switch (a)
       {
          case ACT_SUCCESS:
             return "SUCCESS";
             //break;
          case ACT_ERROR:
             return "ERROR";
             //break;
          case ACT_TRANS_NOT_FOUND:
             return "TRANSFORMATION NOT FOUND";
             //break;
          case ACT_TRANS_FAILED:
             return "TRANSFORMATION FAILED";
             //break;
          case ACT_INPUT_FAILED:
             return "INPUT FAILED";
             //break;
          case ACT_OUPUT_FAILED:
             return "OUTPUT FAILED";
             //break;
          default:
             return "UNKNOWN";
             //break;
       }
    }
```

```
    /**
     * A method used to request that the object perform the action it
describes.  Essentially
     * this means that it should perform each of the ActionItems associated
with this action,
     * such as send an email, etc.
     *
     * This method essentially just calls perform(String) to do the work.
     *
     * @param err    A DtmError object that inidcates the current error state
of the DTM.
     *
     * @return A boolean, indicating whether or not the action was successful.
     */
    public boolean perform(DtmError err)
    {
        return perform("", err);

    }   // perform()


    /**
     * A method used to request that the object perform the action it
describes.  Essentially
     * this means that it should perform each of the ActionItems associated
with this action,
     * such as send an email, etc.
     *
     * @param msg    A message that will be used to 'personlize' the action.
For example, if the action
     *               indicates that an email should be sent, this message may
be used as the email text.
     *
     * @param err    A DtmError object that inidcates the current error state
of the DTM.
     *
     * @return A boolean, indicating whether or not the action was successful.
     */
    public boolean perform(String msg, DtmError err)
    {
        // DEBUG output
        if (DtmConfig.DEBUG > 0)
        {
            System.err.println("DEBUG\tAction started '" + actionName + "'");
            System.err.println("\tError Number : " + err.getNum());
            System.err.println("\tError Action : " + err.getAction());
            System.err.println("\tError Message: " + err.getMsg());
        }

        // now peform each of the Action Items
        for (int ii = 0; ii < actItems.size(); ii++)
        {
            if (!( (ActionItem)actItems.elementAt(ii)).perform(msg, err))
            {
                // some kind of error occurred
                return false;
            }
        }

        // all good
```

```
        return true;

    }   // perform()


    /**
     * Adds a new ActionItem* to the Action object.
     *
     * @param ai     A new ActionItem* object to be added to this Action.
     *
     * @return A boolean, indicating whether or not it was possible to add the
ActionItem* object.
     */
    public boolean addActionItem(ActionItem ai)
    {
        // if the ActionItem is not null, add it to the vector and return true
        if (ai != null)
        {
            actItems.add(ai);
            return true;
        }
        // else we can't add a null object, so return false
        else
        {
            return false;
        }

    }   // addActionItem()

}   // class Action
```

```
/**
 * File:            ActionItem.java
 *
 * @author:         Brian Hay
 *
 * @version:        $Id: ActionItem.java,v 1.2 2004/06/06 04:18:26 bhay Exp $
 *
 */


/* package definition */
package IdactHelpers.Actions;


/* imports - IDACT specific */
import Dtm.Errors.*;


/**
 * Abstract base class for the various ActionItem* classes, which are used to
describe the
 * details of an Action.
 */
public abstract class ActionItem extends Object
{
    abstract public boolean perform(String msg, DtmError err);
}
```

```
/**
 * File:            ActionUnitEmail.java
 *
 * @author:         Brian Hay
 *
 * @version:        $Id: ActionItemEmail.java,v 1.3 2005/04/23 23:30:44 bhay
Exp $
 *
 */


/* package definition */
package IdactHelpers.Actions;


/* imports - IDACT specific */
import Dtm.Errors.*;
import Dtm.Config.*;


/* imports - general */
import javax.mail.*;
import javax.mail.internet.*;
import java.util.*;
import java.util.regex.*;


/**
 * Class to describe and send a specific email message as part of an Action.
 */
public class ActionItemEmail extends ActionItem
{
    /* private constants */
    private final String EMAIL_REG_EXP = "[A-Za-z0-9]+([\\.\\_\\-][A-Za-z0-
9]+)*[\\@][A-Za-z0-9]+([\\.][A-Za-z0-9]+)*";

    /* private fields */
    private String subject;
    private String recipients;
    private String msgBody;


    /**
     * Constructor
     *
     * @param sub       A string containing the subject line to be used.
     *
     *                  This string will be truncated at 100 chars, but can
contain special codes (such as %n, which
     *                  causes the transformation number to be inserted).
     *
     *                  If this parameter is null, then a default subject line
will be used.
     *
     * @param recipList A String containing a comma delimited list of email
recipients.
     *
     *                  Each email will be truncated at 100 characters, and
validated to ensure it fits the format
     *                  [A-Za-z0-9]+[[._][A-Za-z0-9]+]* [@] [A-Za-z0-
```

```
9]+[[._][A-Za-z0-9]+]
     *                   Any address found to be invalid will be discarded.
     *
     *                   At least one valid email must be included in the list.
     *
     * @param msg       A string containing the message to be sent.
     *
     *                   This string can contain special codes (such as %n,
which causes the transformation number
     *                   to be inserted).
     *
     *                   If this parameter is null, then a message will be used.
     */
    public ActionItemEmail(String sub, String recipList, String body)
    {
        String nextAddress;

        // store the subject and message values
        subject = new String(sub);
        msgBody = new String(body);
        recipients = new String(recipList);
    }


    /**
     * This method causes this particular ActionItem to run - in this case,
since this is the
     * Email ActionItem it causes the described email to be sent.
     *
     * @param msg    A string that can be used to 'personalise' the email
message - it may contain information about
     *               a specific error, or the URI of the transformed output
file, for example.
     *
     * @param err    A DtmError object that is used to track any error that
occurs while attempting to run this
     *               ActionItem.
     *
     * @return A boolean indicating whether or not the email was sent
successfully.
     */
    public boolean perform(String msg, DtmError err)
    {
        Properties props = new Properties();
        Session session = Session.getInstance(props, null);

        // create a new email message
        MimeMessage message = new MimeMessage(session);

        // attempt to set the email content
        if (!setContent(message, err))
        {
            return false;
        }

        // attempt to set the email subject line
        if (!setSubject(message, err))
        {
            return false;
        }
```

```
        // attempt to set the address for the sender and receipients
        if (!setAddresses(message, err))
        {
            return false;
        }

        // attempt to send the email
        if (!sendEmail(session, message, err))
        {
            return false;
        }

        // everything went well, so return true
        return true;
    }


    /**
     * A private method that attempts to send the email message.
     *
     * @param session A Session object.
     *
     * @param message A MimeMessage object, which is the message we are
currently building.
     *
     * @param err     The DtmError object, which will be updated in the event
of an error during this operation.
     *
     * @return A boolean indicating whether or not the operation was succesful.
     */
    private boolean sendEmail(Session session, MimeMessage message, DtmError
err)
    {
        try
        {
            // save changes to the email
            message.saveChanges();

            // connect to the SMTP server
            Transport transport = session.getTransport("smtp");
            transport.connect(DtmConfig.SMTP_HOST, DtmConfig.SMTP_USER,
DtmConfig.SMTP_PASS);

            // send the message
            transport.sendMessage(message, message.getAllRecipients());

            // close the connection to the SMTP server
            transport.close();
        }
        catch (MessagingException me)
        {
            // error sending message
            err.set(DtmError.ERR_ACT_MAIL_ME_SEND, Action.ACT_ERROR,
                    ("Failed to send email message."), me );
            return false;
        }

        // all good
        return true;
```

```
    }


    /**
     * A private method that attempt to set the subject for a message.
     *
     * @param message A MimeMessage object, which is the message we are
currently building.
     *
     * @param err     The DtmError object, which will be updated in the event
of an error during this operation.
     *
     * @return A boolean indicating whether or not the operation was succesful.
     */
    private boolean setSubject(MimeMessage message, DtmError err)
    {
        try
        {
            message.setSubject(subject);
        }
        catch (MessagingException me)
        {
            // error setting message subject
            err.set(DtmError.ERR_ACT_MAIL_ME_SUBJ, Action.ACT_ERROR,
                    ("Failed to set subject '" + subject +
                     "' for email message."), me );
            return false;
        }

        // all good
        return true;
    }


    /**
     * A private method that attempts to set the sender and recipients for a
message, based on the recipients Vector.
     *
     * @param message A MimeMessage object, which is the message we are
currently building.
     *
     * @param err     The DtmError object, which will be updated in the event
of an error during this operation.
     *
     * @return A boolean indicating whether or not the operation was succesful.
     */
    private boolean setAddresses(MimeMessage message, DtmError err)
    {
        int ii = 0;     // loop counter

        // set the from address
        try
        {
            message.setFrom(DtmConfig.EMAIL_FROM);
        }
        catch (MessagingException me)
        {
            // error setting message sender
            err.set(DtmError.ERR_ACT_MAIL_ME_SNDR, Action.ACT_ERROR,
                    ("Failed to set message sender '" + DtmConfig.EMAIL_FROM +
```

```
                    "' for email message"), me );
            return false;
        }

        // Vector of recipient addresses
        Vector r = new Vector(3, 5);

        // populate the recipient list
        if (!initRecipients(r))
        {
            // we don't have a valid list of recipients, so there's nothing to
do
            return false;
        }

        // attempt to add each of the recipients to the message
        try
        {
            // convert the recipient address vector to an array
            Address ra[] = new Address[r.size()];
            r.toArray(ra);
            // now set the addresses using this new recipient address array.
            message.setRecipients( Message.RecipientType.TO, ra );
        }
        catch (MessagingException me)
        {
            // error setting message recipients
            err.set(DtmError.ERR_ACT_MAIL_ME_RECIP, Action.ACT_ERROR,
                    ("Failed to add recipient address '" +
((InternetAddress)r.elementAt(ii)).getAddress() +
                     "' to email message"), me );
            return false;
        }

        // all good
        return true;
    }


    /**
     * A private method that attempts to set the content for a message.
     *
     * @param message A MimeMessage object, which is the message we are
currently building.
     *
     * @param err     The DtmError object, which will be updated in the event
of an error during this operation.
     *
     * @return A boolean indicating whether or not the operation was succesful.
     */
    private boolean setContent(MimeMessage message, DtmError err)
    {
        try
        {
            message.setContent(msgBody, "text/plain");
            //message.setText("Hello");
        }
        catch (MessagingException me)
        {
            // error setting message content
```

```
            err.set(DtmError.ERR_ACT_MAIL_ME_CONT, Action.ACT_ERROR,
                    ("Failed to set content in email message."), me );
            return false;
        }

        // all good
        return true;
    }


    /**
     * A private method that initializes the recipients vector.
     *
     * @param recipList A String containing a comman-delimited list of email
recipeints.
     *
     * @return A boolean, indicating whether or not there was at least one
valid email address.
     */
    private boolean initRecipients(Vector r)
    {
        // a String to hold the next address
        String nextAddress;

        // create a pattern for matching email addresses
        Pattern p = Pattern.compile(EMAIL_REG_EXP);

        // populate the vector with email addresses;
        String tmpList = new String( recipients.trim() );
        while ( !tmpList.equals("") )
        {
            // find the location of the first comma
            int commaLoc = tmpList.indexOf(",");

            // if there is no comma left in the list, then the whole string is
our last address
            if (commaLoc < 0)
            {
                nextAddress = new String(tmpList);
                tmpList = new String("");
            }
            // else grab the chars up to the comma as the next address, and
keep the remainder as
            // the rest of the list.
            else
            {
                nextAddress = tmpList.substring(0, commaLoc).trim();
                tmpList = tmpList.substring(commaLoc + 1).trim();
            }

            // check that the next address is in a valid format, using our
pattern
            Matcher m = p.matcher(nextAddress);
            if ( (nextAddress.length() <= 100) && (m.matches()) )
            {
                try
                {
                    r.add(new InternetAddress(nextAddress));
                }
                catch (AddressException ae)
```

```
                {
                    if (DtmConfig.DEBUG > 0)
                    {
                        System.err.println("DEBUG\t" +
                                           "Failed to set recipient in email
message." +
                                           "\tstack trace:\n");
                        ae.printStackTrace();
                    }
                }
            }
            // else DEBUG output
            else if (DtmConfig.DEBUG > 0)
            {
                System.err.println("DEBUG\tActionItemEmail(): Invalid email
address");
                if (nextAddress.length() > 100)
                {
                    System.err.println("\tEmail address too long");
                }
                else
                {
                    System.err.println("\tDoes not fit pattern ('" +
nextAddress + "')");
                }
            }
        }
        // return true if there was at least one valid email address, and false
otherwise.
        return (r.size() > 0);
    }

}
```

```
/*
 * Copyright 2004 Brian Hay (IDACT)
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

/**
 * An abstract class used at the basis for the various other
XmlStructFormatDesc*
 *   classes, which allow the manner in which 'from' elements, functions, and
strings,
 *   for example, as used to populate a given 'to' element.
 *
 * @author Brian Hay
 * @version $Id: XmlStructFormatDesc.java,v 1.2 2004/10/31 22:11:28 bhay Exp $
 */
public abstract class XmlStructFormatDesc
{
    /** public static fields that define the various types that
     *  can be used in the format descriptions.
     */
    public static final int TYPE_UNDEF    = -1;
    public static final int TYPE_ELEMENT  = 0;
    public static final int TYPE_STRING   = 1;
    public static final int TYPE_TEMPLATE = 2;
    //public static final int TYPE_FUNCTION = 3;

    // an int variable used to store the type of each particular object
    private int type;


    /**
     * This public accessor method get the type of the particular object, which
     * should be one of the values defined as TYPE_* static members of this
class.
     *
     * @return An int representing the type of this object.
     */
    public int getType()
    {
       return type;
    }

    /**
     * This public mutator method allows the type for this object to be
specified.
     *
     * @param newType The type for this particular object.  This should
```

```
correspond to one of
     *                the defined static TYPE_* constants in this class.
     */
    public void setType(int newType)
    {
       type = newType;
    }
}
```

```
/*
 * Copyright 2004 Brian Hay (IDACT)
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

/**
 * An class used to record element references for use in populating a given
'to'
 *   element.
 *
 * @author Brian Hay
 *
 * @version $Id: XmlStructFormatDescElement.java,v 1.2 2004/10/31 22:11:28 bhay
Exp $
 */
public class XmlStructFormatDescElement extends XmlStructFormatDesc
{
    // private int used to store the index in the link's fromNodes Vector that
    // corresponds to the element to use.
    int elementId;


    /**
     * Constructor for the XmlStructFormatDescElement class, which sets the type
     * to TYPE_ELEMENT, and sets the element index value.
     *
     * @param newElementId
     *               The index value of the element in the link's fromNodes
Vector.
     */
    public XmlStructFormatDescElement(int newElementId)
    {
        super.setType(XmlStructFormatDesc.TYPE_ELEMENT);
        elementId = newElementId;
    }

    /**
     * Accessor method for the element index.
     *
     * @return An int representing the element's index in the link's fromNodes
Vector.
     */
    public int getElementId()
    {
        return elementId;
    }
}
```

```
/*
 * Copyright 2004 Brian Hay (IDACT)
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

/**
 * An class used to record string constants for use in populating a given 'to'
 *   element.
 *
 * @author Brian Hay
 *
 * @version $Id: XmlStructFormatDescString.java,v 1.2 2004/10/31 22:11:28 bhay
Exp $
 */
public class XmlStructFormatDescString extends XmlStructFormatDesc
{
   // private object used to store the string value
   private String str;


   /**
    * Constructor for the XmlStructFormatDescString class, which sets the type
    * to TYPE_STRING, and sets the value of the string.
    *
    * @param newString The new string value to be stored.
    */
   public XmlStructFormatDescString(String newString)
   {
      super.setType(XmlStructFormatDesc.TYPE_STRING);
      str = new String(newString);
   }

   /**
    * Accessor method for the string stored in this object.
    *
    * @return A copy of the string stored in this object.
    */
   public String getString()
   {
      return new String(str);
   }
}
```

```
/*
 * Copyright 2004 Brian Hay (IDACT)
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

import java.util.Vector;

/**
 * An class used to record template references for use in populating a given
'to'
 *   element.
 *
 * @author Brian Hay
 *
 * @version $Id: XmlStructFormatDescTemplate.java,v 1.3 2004/11/03 06:30:24
bhay Exp $
 */
public class XmlStructFormatDescTemplate extends XmlStructFormatDesc
{
   // String object to hold the template name
   private String templateName;

   // Vector to hold references to the parameters for this template,
   // which may be of any of the types defined in the XmlStructFormatDesc
   // class TYPE_* constants.
   private Vector params;


   /**
    * Constructor for the fromDescTemplate class, which sets the type
    * to TYPE_TEMPLATE, sets the template name, and creates an empty
    * parameter list.
    *
    * @param newTemplateName
    *               A String containing the name of the template.
    */
   public XmlStructFormatDescTemplate(String newTemplateName)
   {
      this( newTemplateName, new Vector() );
   }


   /**
    * Constructor for the fromDescTemplate class, which sets the type
    * to TYPE_TEMPLATE, sets the template name, and creates a parameter
    * list based on the Vector provided.
    *
```

```
     * @param newTemplateName
     *                  A String containing the name of the template.
     * @param newParams A Vector object containing the parameter list for this
template.
     */
    public XmlStructFormatDescTemplate(String newTemplateName, Vector newParams)
    {
       super.setType(XmlStructFormatDescTemplate.TYPE_TEMPLATE);
       templateName = new String(newTemplateName);
       params = (Vector)newParams.clone();
    }



    /**
     * Accessor method for the template name.
     *
     * @return A String object containing the template's index name.
     */
    public String getName()
    {
       return new String(templateName);
    }


    /**
     * Accessor method for the size (length) of the parameter list.
     *
     * @return An int representing the number of items in the parameter list.
     */
    public int getNumParams()
    {
       return params.size();
    }


    /**
     *
     * Accessor method for individual parameter list items.
     *
     * @param index
     *
     * @return The XmlStructFormatDesc object at the specified index in the
     *         parameter list.
     */
    public XmlStructFormatDesc getParam(int index)
    {
       if ( (index < 0) || (index >= params.size() ) )
       {
          return null;
       }

       return (XmlStructFormatDesc)params.elementAt(index);
    }

}
```

```
/*
 * Copyright 2004 Brian Hay (IDACT)
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

// import the Vector class
import java.util.Vector;


/**
 * This class is used in to define links between sections of the 'from' and
'to'
 * XML structure in the XSLT Generator.  It allows for many-to-one links to be
 * defined, with an associated transformation/conversion.
 *
 * @author Brian Hay
 *
 * @version $Id: XmlStructLink.java,v 1.7 2005/01/03 08:09:36 bhay Exp $
 */
public class XmlStructLink
{
   // reference to the node in the 'to' structure for this link
   XmlStructNode toNode;

   // Vector of references to the nodes in the 'from' structure for this link
   Vector fromNodes;

   // sort operators for each of the nodes in the 'from' structure for this
link
   XmlStructSort fromSort;

   // description of how to use the 'from' fields to create the 'to' field
   XmlStructFormatDescTemplate formatDesc;


   /**
    * Default constructor
    */
   public XmlStructLink()
   {
      this(null);
   }


   /**
    * Constructor
    *
```

```
    * @param newToNode An XmlStructNode object that references the node in the
'to' structure
    *                  for this link.
    */
   public XmlStructLink(XmlStructNode newToNode)
   {
      toNode = newToNode;
      fromNodes = new Vector();
      fromSort = new XmlStructSort();
      formatDesc = null;
   }


   public void setFormatDesc(XmlStructFormatDescTemplate newFormatDesc)
   {
      formatDesc = newFormatDesc;
   }

   public XmlStructFormatDescTemplate getFormatNode()
   {
      return formatDesc;
   }


   /**
    * This method adds a new node in the 'from' structure to this link
    * relationship.
    *
    * @param newFromNode
    *               The new link in the 'from' structure to be added to this
link
    *               relationship.
    *
    * @return A boolean indicating the success (true) or failure (false) of the
    *         operation.
    *
    */
   public boolean addFromNode(XmlStructNode newFromNode)
   {
      return addFromNode( newFromNode, new XmlStructSort() );
   }


   /**
    * This method adds a new node in the 'from' structure to this link
    * relationship.
    *
    * @param newFromNode
    *               The new link in the 'from' structure to be added to this
link
    *               relationship.
    *
    * @param newSort
    *               An XmlStructSort object that describes the nodes in the
'from' structure
    *               that will be used to sort this node.
    *
    * @return A boolean indicating the success (true) or failure (false) of the
    *         operation.
    *
```

```
     */
    public boolean addFromNode(XmlStructNode newFromNode, XmlStructSort newSort)
    {
       // check for a null object
       if ( (null != newFromNode) && (null != newSort) )
       {
          // check that this node does not already exist in the vector
          for (int ii = 0; ii < fromNodes.size(); ii++)
          {
             if ( newFromNode == (XmlStructNode)fromNodes.elementAt(ii) )
             {
                // node already exists, so nothing to do
                return true;
             }
          }
          // add the new 'from' node now, together with the sort object
          fromNodes.add(newFromNode);
       }
       else
       {
          // null object, so error
          System.err.println("ERROR:\tXmlStructLink::addFromNode()\n\tAttempted
to add null node!");
          return false;
       }

       // all OK, so return true
       return true;

    }  // addFromNode()


    /**
     * This method removes a node in the 'from' structure from this link
     * relationship.
     *
     * @param delFromNode
     *               The new link in the 'from' structure to be removed from
this
     *               link relationship.
     *
     * @return A boolean indicating the success (true) or failure (false) of the
     *         operation.
     *
     */
    public boolean removeFromNode(XmlStructNode delFromNode)
    {
       // check for a null object
       if ( null != delFromNode)
       {
          // locate the node's index so we can delete from both Vectors
          int index = fromNodes.indexOf(delFromNode);

          // if the element exists in the Vector
          if ( (index >= 0) && (index < fromNodes.size() ) )
          {
             // attempt to delete the node and the sort
             fromNodes.removeElementAt(index);
             return true;
          }
```

```
         else
         {
            // the node we're trying to delete does not exist

   System.err.println("ERROR:\tXmlStructLink::removeFromNode()\n\tAttempted to
delete noexistant node!");
            return false;
         }
      }
      else
      {
         // null object, so error

   System.err.println("ERROR:\tXmlStructLink::removeFromNode()\n\tAttempted to
delete noexistant node!");
         return false;
      }

   }  // removeFromNode()



   /**
    * This method cleans up the link before deletion.
    */
   public void cleanup()
   {
      // delete all links to 'from' nodes
      for (int ii = getNumFromNodes() - 1; ii >= 0; ii--)
      {
         ((XmlStructNode)(fromNodes.elementAt(ii))).setLink(null);
         fromNodes.removeElementAt(ii);
      }

      // delete the link to the 'to' node
      toNode.setLink(null);
      toNode = null;
   }



   /**
    * This method gets the number of 'from' nodes associated with this link
    * relationship.
    *
    * @return An int value, indicating the number of 'from' nodes ssociated
with this
    *         link relationship.
    *
    */
   public int getNumFromNodes()
   {
      return fromNodes.size();

   }  // getNumFromNodes()



   /**
    * This method gets the 'to' node associated with this link relationship.
```

```
     *
     * @return An XmlStruct reference to the 'to' node associated with this link
     *         relationship.
     *
     */
    public XmlStructNode getToNode()
    {
       return toNode;

    }  // getToNode()



    /**
     * This method gets one of the 'from' nodes associated with this link
     * relationship.
     *
     * @param nodeNum The index (zero-based) of the 'from' node to get.
     *
     * @return An XmlStruct reference to the requested 'from' node associated
with this
     *         link relationship, or null if the index provided was out of
bounds.
     *
     */
    public XmlStructNode getFromNode(int nodeNum)
    {
       if ( (nodeNum >= 0) && (nodeNum < fromNodes.size() ) )
       {
          return (XmlStructNode)fromNodes.elementAt(nodeNum);
       }
       else
       {
          return null;
       }

    }  // getFromNode()


    /**
     * This method gets one of the sort for a 'from' node associated with this
link
     * relationship.
     *
     * @return An XmlStructSort reference to the requested sort for a 'from'
node
     *         associated with this link relationship, or null if the index
provided
     *         was out of bounds.
     *
     */
    public XmlStructSort getSort()
    {
       return fromSort;

    }  // getFromSort()

}  // class XmlStructLink
```

```
/*
 * Copyright 2004 Brian Hay (IDACT)
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

// general imports
import java.util.Vector;


/**
 * This class is used to build the structure of a valid XML document from
 * a XSD file.  Objects of this class represent nodes in a hierarchical
 * structure, with links to parent and child nodes, and also with links to
 * nodes in other strucutres (so that element equivalences can be defined).
 *
 * @author Brian Hay
 *
 * @version $Id: XmlStructNode.java,v 1.6 2004/10/31 22:11:28 bhay Exp $
 *
 */
public class XmlStructNode
{
   // a vector of subnodes
   private Vector subnodes;

   // a reference to the parent node of this node
   private XmlStructNode parentNode;

   // data type for this node
   private int nodeType;

   // node name
   private String nodeName;

   // link reference
   private XmlStructLink link;

   // flag to indicate if descendant nodes have links
   private boolean descendantLinks;

   // XPath from the root node to this node
   String pathToNode;

   // node depth, starting with the root node at level 1
   int nodeDepth;

   // default constructor
```

```
public XmlStructNode()
{
   this("", null);
}

// constructor
public XmlStructNode(String newName)
{
   this(newName, null);
}

// constructor
public XmlStructNode(String newName, XmlStructNode newParentNode)
{
   nodeName = newName;
   parentNode = newParentNode;
   link = null;
   subnodes = new Vector();
   descendantLinks = false;
   pathToNode = new String("");
   nodeDepth = -1;
}

// set the descendant links flag
public void setDescendantLinks(boolean dl)
{
   descendantLinks = dl;
}

// get the descendant links flag
public boolean getDescendantLinks()
{
   return descendantLinks;
}

// set the node depth
public void setNodeDepth(int depth)
{
   nodeDepth = depth;
}

// get the depth
public int getNodeDepth()
{
   return nodeDepth;
}

// set the path
public void setPath(String newPath)
{
   pathToNode = new String(newPath);
}

// get the path
public String getPath()
{
   return new String(pathToNode);
}

// set the node name
```

```
    public void setNodeName(String newName)
    {
       nodeName = newName;
    }

    // get the node name
    public String getNodeName()
    {
       return nodeName;
    }

    // get the number of subnodes
    public int getNumSubnodes()
    {
       return subnodes.size();
    }

    // get the parent node for this node
    public XmlStructNode getParentNode()
    {
       return parentNode;
    }

    // get the link node for this node
    public XmlStructLink getLink()
    {
       return link;
    }

    // get one of the subnodes
    public XmlStructNode getSubNode(int n)
    {
       if ( (n >= 0) && (n < subnodes.size() ) )
       {
          return( (XmlStructNode)subnodes.elementAt(n) );
       } else
       {
          return null;
       }
    }

    // add a new subnode
    public boolean addSubNode(XmlStructNode newNode)
    {
       return subnodes.add(newNode);
    }

    // set the parent node
    public void setParentNode(XmlStructNode newParentNode)
    {
       parentNode = newParentNode;
    }

    // set the link node
    public void setLink(XmlStructLink newLink)
    {
       link = newLink;
    }
}
```

```
/*
 * Copyright 2004 Brian Hay (IDACT)
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

// import the Vector class
import java.util.Vector;


/**
 * This class is used in to define the sorts to be applied when using a 'from'
 * node.
 *
 * @author Brian Hay
 *
 * @version $Id: XmlStructSort.java,v 1.2 2004/10/31 22:11:28 bhay Exp $
 */
public class XmlStructSort
{
   static final String ORDER_ASCEND  = "ascending";
   static final String ORDER_DESCEND = "descending";

   // Vector for each field to be sorted
   Vector nodes, orders;

   public XmlStructSort()
   {
      nodes = new Vector();
      orders = new Vector();
   }

   public boolean addSort(XmlStructNode sortNode, String order)
   {
      return ( (nodes.add(sortNode) ) && ( orders.add( new String(order) ) ) );
   }

   public int getNumSorts()
   {
      if (nodes.size() == orders.size())
      {
         return nodes.size();
      }
      else
      {
         System.err.println("ERROR:\tXmlStructSort::getNumSorts()\n\tVector
size mismatch!");
         return -1;
```

```
        }
    }

    public XmlStructNode getSortNode(int sortNum)
    {
        if ( (sortNum >= 0) && (sortNum < nodes.size() ) )
        {
            return (XmlStructNode)nodes.elementAt(sortNum);
        }
        else
        {
            return null;
        }
    }

    public String getSortOrder(int sortNum)
    {
        if ( (sortNum >= 0) && (sortNum < orders.size() ) )
        {
            return (String)orders.elementAt(sortNum);
        }
        else
        {
            return null;
        }
    }

}   // class XmlStructSort
```

```
/*
 * Copyright 2004 Brian Hay (IDACT)
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

// imports for the XML processing
import org.apache.xerces.parsers.DOMParser;
import org.w3c.dom.Document;
import org.w3c.dom.DocumentType;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

// imports for the file output
import java.io.*;

// imports for the Vector and Set functionality
import java.util.Vector;
import java.util.Set;
import java.util.HashSet;
import java.util.Iterator;

// import for the database SQL interaction
import java.sql.*;



/**
 * A prototype of an XSLT Generator.
 *
 * @author Brian Hay
 *
 * @version $Id: XsltGenerator.java,v 1.17 2005/01/03 08:09:36 bhay Exp $
 *
 */
public class XsltGenerator
{
    /**
     * A constant for the debug level ( a value greater than 0 will cause debug
     * messages to be displayed).
     */
    private final int DEBUG = 0;


    /**
     * A String that will be used as temporary storage for the XSLT as it is
     * generated, prior to it being written to the actual XSLT file.
```

```
     */
    String xsltString;


    /**
     * A Set that contains all of the XSLT components to be retreived from the
database
     * and written to the XSLT output
     */
    private Set xsltComponents = new HashSet();

    private String xsltFile;                // string that contains the XSLT
output filename
    private XmlStructNode fromStructRoot;       // root node for the 'from'
format structure
    private XmlStructNode toStructRoot;         // root node for the 'to' format
structure

    public final static int TO_STRUCT   = 0;
    public final static int FROM_STRUCT = 1;

    /**
     * Constructor that accepts three filenames (input format, output format,
and
     * XSLT output).  An attempt is made to open the format files, and build
their
     * structure.
     *
     * @param fromFile A String containing the name of the input format file.
     * @param toFile   A String containing the name of the output format file.
     * @param xsltFile A String containing the name of the XSLT output file.
     *
     * @exception Exception An Exception object that contains a message
indicating the error, if
     *                      an error condition arises.
     */
    public XsltGenerator(String fromFile, String toFile, String outXsltFile)
throws Exception
    {
        xsltString = new String("");
        xsltFile = new String(outXsltFile);

        // valid parameter checks
        if ( (null == fromFile) || (null == toFile) || (null == xsltFile) )
        {
            throw new Exception("Null filename parameters passed to
XsltGenerator");
        }
        if ( (fromFile.length() < 1) || (toFile.length() < 1) ||
(xsltFile.length() < 1) )
        {
            throw new Exception("Empty filename parameters passed to
XsltGenerator");
        }

        // build the 'from' structure
        if ( null == ( fromStructRoot = buildStructure(fromFile) ) )
        {
            if (DEBUG > 0) System.err.println("ERROR:\tXsltGenerator()\n\tFailed
to create structure for 'from' XSD file!\n\tfilename='"+fromFile+"'");
```

```
            throw new Exception("Failed to create 'from' structure");
         }

         // build the 'to' structure
         if ( null == ( toStructRoot = buildStructure(toFile) ) )
         {
            if (DEBUG > 0) System.err.println("ERROR:\tXsltGenerator()\n\tFailed
to create structure for 'to' XSD file!\n\tfilename='"+toFile+"'");
            throw new Exception("Failed to create 'to' structure");
         }

      }

      /**
       * This method displays the 'from' structure, if one has been defined.
       *
       * @return A boolean, which is false if the 'from' structure has not yet
been defined,
       *         and true otherwise.
       */
      public boolean printFromStruct()
      {
         if (null == fromStructRoot)
         {
            return false;
         }
         printStruct(fromStructRoot);
         return true;
      }


      /**
       * This method displays the 'to' structure, if one has been defined.
       *
       * @return A boolean, which is false if the 'to' structure has not yet been
defined,
       *         and true otherwise.
       */
      public boolean printToStruct()
      {
         if (null == toStructRoot)
         {
            return false;
         }
         printStruct(toStructRoot);
         return true;
      }


      /**
       * This method checks to determine if all of the requirements necessary to
       * allow XSLT generation have been met.
       *
       * @return A boolean, with true indicating the XSLT generation is possible,
and false
       *         indicating that it is not possible.
       */
      public boolean readyForXslt()
      {
         return checkLinks(toStructRoot);
```

```
   }


   /**
    * Allow a link to be added between one or more input ('from') fields and a
    * given output ('to') field.
    *
    * @param toField    The node to link to in the output structure.
    * @param fromFields A Vector of Vectors, each of which describes the path
to a node in the
    *                   input structure that will be part of the link.
    *
    * @exception Exception
    */
   public void addNewLink(Vector toField, Vector fromFields) throws Exception
   {
      XmlStructNode toNode = null, fromNode = null;
      boolean invalidFields = false;

      if ( (null == toField) || (null == fromFields) )
      {
         if (DEBUG > 0) System.err.print("ERROR:\taddNewLink()\n\tNull vectors
passed as parameters\n");
         throw new Exception("Null field vectors found");
      }

      // attempt to locate the 'to' field and handle case where field is not
valid
      try
      {
         toNode = findNode(toStructRoot, toField);
      }
      catch (Exception e)
      {
         if (DEBUG > 0)
         {
            System.err.print("ERROR:\taddNewLink()\n\tInvalid 'to' field\n");
            for (int ii = 0; ii < toField.size(); ii++)
            {
               System.out.print(((Integer)toField.elementAt(ii)).intValue() +
" ");
            }
            System.out.println();
         }
         invalidFields = true;
      }

      // for each 'from' field
      for (int jj = 0; jj < fromFields.size(); jj++ )
      {
         // attempt to locate the 'from' field and handle case where field is
not valid
         try
         {
            fromNode = findNode(fromStructRoot,
(Vector)fromFields.elementAt(jj));
         }
         catch (Exception e)
         {
            if (DEBUG > 0)
```

```
            {
               System.err.print("ERROR:\taddNewLink()\n\tInvalid 'from'
field\n");
               Vector v = (Vector)fromFields.elementAt(jj);
               for (int ii = 0; ii < v.size(); ii++)
               {
                  System.out.print(((Integer)v.elementAt(ii)).intValue() + "
");
               }
               System.out.println();
            }
            invalidFields = true;
         }

         if (!invalidFields)
         {
            // create the link if necessary
            if (null == toNode.getLink())
            {
               toNode.setLink( new XmlStructLink(toNode) );
            }

            // check that we haven't already added this field
            boolean addedAlready = false;
            for (int ii = 0; (ii < toNode.getLink().getNumFromNodes()) &&
!addedAlready; ii++)
            {
               if (fromNode == toNode.getLink().getFromNode(ii))
               {
                  addedAlready = true;
               }
            }

            // add the link between the nodes if necessary
            if (!addedAlready)
            {
               toNode.getLink().addFromNode(fromNode);
            }
            else
            {
               if (DEBUG > 0)
System.err.print("DEBUG:\taddNewLink()\n\tSkipped duplicate field " +
fromNode.getNodeName() + "\n");
            }
         }
      }

      if (invalidFields)
      {
         throw new Exception("Invalid fields found while adding new link");
      }
   }


   /**
    * This method deletes an entire link for a given node in the 'to'
structure.
    *
    * @param toField A Vector that describes the path to the 'to' field for
which the link
```

```
     *                should be deleted.
     *
     * @exception Exception
     */
    public void deleteEntireLink(Vector toField) throws Exception
    {
       // get the 'to' node
       XmlStructNode toNode = findNode(toStructRoot, toField);

       // get the link object
       XmlStructLink link = toNode.getLink();
       if (null != link)
       {
          link.cleanup();
       }
       else
       {
          // no link to delete
          if (DEBUG > 0) System.err.println("DEBUG:\tdeleteEntireLink()\n\tNo
link to delete for node '" + toNode.getNodeName() + "'");
       }
    }


    /**
     * This method deletes part of a link for a given node in the 'to'
structure.
     *
     * @param toField   A Vector that describes the path to the 'to' field for
which the link
     *                  should be deleted.
     * @param fromField A Vector that describes the path to the 'from' field for
which the link
     *                  should be deleted.
     *
     * @exception Exception
     */
    public void deleteLink(Vector toField, int fromNodeChoice) throws Exception
    {
       // get the 'to' node
       XmlStructNode toNode = findNode(toStructRoot, toField);

       // get the link object
       XmlStructLink link = toNode.getLink();
       if (null != link)
       {
          XmlStructNode fromNode = link.getFromNode(fromNodeChoice);
          if (!link.removeFromNode(fromNode))
          {
             if (DEBUG > 0) System.err.println("DEBUG:\tdeleteLink()\n\tFailed
to delete link from node '" + fromNode.getNodeName() + "' to node '" +
toNode.getNodeName() + "'");
             throw new Exception("Failed to delete link from node '" +
fromNode.getNodeName() + "' to node '" + toNode.getNodeName() + "'");
          }
       }
       else
       {
          // no link to delete
          if (DEBUG > 0) System.err.println("DEBUG:\tdeleteLink()\n\tNo link to
```

```
delete for node '" + toNode.getNodeName() + "'");
      }

      // delete link structure if there are no remaining links
      if (link.getNumFromNodes() < 1)
      {
         link.cleanup();
      }
   }



   /**
    * This method creates and returns a Vector populated with the names of the
    * 'from' nodes linked to a given 'to' node.
    *
    * @param toNodePath A Vector representing the path to the 'to' node.
    *
    * @return A Vector containing the names of the 'from' nodes linked to the
specified
    *         'to' node.
    * @exception Exception
    */
   public Vector getLinkedNodeNames(Vector toNodePath) throws Exception
   {
      Vector linkedNodeNames = new Vector();

      XmlStructNode toNode = findNode(toStructRoot, toNodePath);
      XmlStructLink link = toNode.getLink();

      if (null != link )
      {
         for (int ii = 0; ii < link.getNumFromNodes(); ii++)
         {
            linkedNodeNames.add(link.getFromNode(ii).getNodeName());
         }
      }
      return linkedNodeNames;
   }



   /**
    * Attempts to find all subnodes for a given node
    *
    * @param nodePath A Vector containing the path to the parent node.
    * @param ioStruct An int that determines whether the parent node is in the
input or output
    *                 structure.  This values should be one of FROM_STRUCT or
TO_STRUCT.
    *
    * @return A Vector containing the names of the subnodes.  The Vector
returned is empty
    *         in the case that the node has no subnodes.
    *
    * @exception Exception Thrown in the case of error.
    */
   public Vector getSubNodeNames(Vector nodePath, int ioStruct) throws
Exception
   {
```

```
       Vector subNodeNames = new Vector();

       // get the appropriate root node
       XmlStructNode structRoot;
       if (ioStruct == FROM_STRUCT)
       {
          structRoot = fromStructRoot;
       }
       else if (ioStruct == TO_STRUCT)
       {
          structRoot = toStructRoot;
       }
       else
       {
          if (DEBUG > 0)
System.err.println("ERROR:\tgetSubNodeNames()\n\tioStruct parameter invalid
("+ioStruct+")");
          throw new Exception("Invalid struct choice");
       }

       // build the vector of subnode names
       if ( (null == nodePath) || (nodePath.size() < 1 ) )
       {
          subNodeNames.add( structRoot.getNodeName() );
       }
       else
       {
          // attempt to find the node in question
          XmlStructNode n = findNode(structRoot, nodePath);

          // now find all of the subnode names, if any exist
          if (n.getNumSubnodes() < 1)
          {
             subNodeNames = null;
          }
          else
          {
             for (int ii = 0; ii < n.getNumSubnodes(); ii++)
             {
                subNodeNames.add(n.getSubNode(ii).getNodeName());
             }
          }
       }
       return subNodeNames;
    }



    /**
     * Finds a node given a start node and a path.
     *
     * @param node     The node at which to start the search (usually the root
node of either
     *                 the input or output structure).
     * @param nodePath A Vector describing the path from the search node to the
desired node.
     *
     * @return An XmlStructNode reference to the node that was found.
     * @exception Exception
     */
```

```
   private XmlStructNode findNode(XmlStructNode node, Vector nodePath) throws
Exception
   {
      if (null == node)
      {
         if (DEBUG > 0) System.err.println("ERROR:\tfindNode()\n\tnode
parameter is null");
         throw new Exception("No node defined");
      }

      for (int ii = 1; ii < nodePath.size(); ii++)
      {
         node = node.getSubNode( ((Integer)nodePath.elementAt(ii)).intValue()
);
         if (null == node)
         {
            if (DEBUG > 0) System.err.println("ERROR:\tfindNode()\n\tFailed to
get subnode at depth "+ii);
            throw new Exception("No such subnode");
         }
      }
      return node;
   }


   /**
    */
   public void showLinks()
   {
      printLinks(toStructRoot);
   }



   public boolean writeXsltDoc()
   {
      // attempt to build the XSLT document
      if ( !buildXslt(toStructRoot, fromStructRoot) )
      {
         System.err.println("ERROR:\tstartProcess()\n\tFailed to build XSLT
document!");
         return false;
      }

      // attempt to write the XSLT file
      if ( !writeXsltFile(xsltFile) )
      {
         System.err.println("ERROR:\tstartProcess()\n\tFailed to write XSLT
file!");
         return false;
      }

      return true;
   }



   private boolean buildFormatsManual(XmlStructNode xsn, String indent)
   {
```

```
      // build the format for this node
      try
      {
         System.out.println("\n" + indent + "Build format for 'to' element " +
xsn.getNodeName() );
         xsn.getLink().setFormatDesc( getParams(xsn.getLink(), indent, true)
);
      }
      catch (IOException ioe)
      {
         System.err.println("ERROR:\tbuildFormatsManual()\n\tIOException for
node " + xsn.getNodeName() + "!");
         return false;
      }

      // recursively build the format for each sub node
      for (int ii = 0; ii < xsn.getNumSubnodes(); ii++)
      {
         if (!buildFormatsManual(xsn.getSubNode(ii), (indent + "  ")))
         {
            return false;
         }
      }
      return true;
   }


   public XmlStructFormatDescTemplate getParams(XmlStructLink xsl, String
indent, boolean isStart) throws IOException
   {
      String name, line;
      int type;
      Vector params = new Vector();
      BufferedReader keyboard = new BufferedReader( new InputStreamReader(
System.in ) );

      if (isStart)
      {
         name = "CONCAT";
      }
      else
      {
         System.out.print(indent + "Enter template name: ");
         name = keyboard.readLine();
      }

      // get the parameters
      do
      {
         System.out.println(indent + XmlStructFormatDesc.TYPE_ELEMENT + "-
ELEMENT\t" +
                            XmlStructFormatDesc.TYPE_STRING + "-STRING\t" +
                            XmlStructFormatDesc.TYPE_TEMPLATE + "-TEMPLATE");
         System.out.print(indent + "Enter param type: ");
         line = keyboard.readLine();
         if ( (null != line) && (line.length() > 0) )
         {
            type = Integer.parseInt(line);
            switch (type)
            {
```

```
                case XmlStructFormatDesc.TYPE_ELEMENT:
                   System.out.println();
                   for (int ii = 0; ii < xsl.getNumFromNodes(); ii++)
                   {
                      System.out.println(indent + ii + ": " +
xsl.getFromNode(ii).getNodeName());
                   }
                   System.out.print(indent + "Enter element ID: ");
                   String elemIdStr = keyboard.readLine();
                   int elemId = Integer.parseInt(elemIdStr);
                   if ( (elemId >= 0) && (elemId < xsl.getNumFromNodes() ) )
                   {
                      params.add( new XmlStructFormatDescElement(elemId) );
                   }
                   else
                   {
                      System.out.println(indent + "Invalid element ID!");
                   }
                   break;
                case XmlStructFormatDesc.TYPE_STRING:
                   System.out.print(indent + "Enter string: ");
                   String str = keyboard.readLine();
                   params.add( new XmlStructFormatDescString(str) );
                   break;
                case XmlStructFormatDesc.TYPE_TEMPLATE:
                   params.add( getParams( xsl, (indent + "  "), false ) );
                   break;
                default:
                   System.out.println(indent + "Invalid parameter type!");
                   break;
             }
          }
       } while ( (null != line) && (line.length() > 0) );

       XmlStructFormatDescTemplate f = new XmlStructFormatDescTemplate(name,
params);

       return f;
   }


   /**
    * This method checks each of the nodes in the 'to' structure to ensure that
    * they each have a corresponding link, and returns false if any are found
    * that do not.
    *
    * @param xsn    A node in the 'to' structure
    *
    * @return A boolean indicating that the node and all subnodes have links
(true), or
    *         that one or more wwere found without links (false).
    */
   private boolean checkLinks(XmlStructNode xsn)
   {
      boolean retVal = true;

      // handle the case of an unmatched node
      if ( null == xsn.getLink() )
      {
         // can't have an unlinked node, so show error message
```

```
            if (DEBUG > 0) System.out.println("DEBUG:\tcheckLinks()\n\tUnlinked
node: " + xsn.getNodeName() );
            retVal = false;
         }

         for (int ii = 0; ii < xsn.getNumSubnodes(); ii++)
         {
            retVal = checkLinks(xsn.getSubNode(ii)) & retVal;
         }

         return retVal;
      }



      /**
       * This method attempts to determine the structure of an XML file, based
       * on an XSD document.
       *
       * @param xsdFile The name of the XSD file to be used to create the
structure/
       *
       * @return An XMLStructNode object that is the head of the structure, or
null in the
       *         case of failure to build the structure.
       */
      private XmlStructNode buildStructure(String xsdFile)
      {
         DOMParser parser = new DOMParser();

         // attempt to open the XSD document in the parser
         try
         {
            parser.parse(xsdFile);
         }
         catch (org.xml.sax.SAXException se)
         {
            if (DEBUG > 0) System.err.println("ERROR:\tbuildStructure()\n\tFailed
to open XSD file!\n\torg.xml.sax.SAXException\n\t" + se.getMessage() );
            return null;
         }
         catch (java.io.IOException ioe)
         {
            if (DEBUG > 0) System.err.println("ERROR:\tbuildStructure()\n\tFailed
to open XSD file!\n\tjava.io.IOException\n\t" + ioe.getMessage() );
            return null;
         }

         // get a root node for this structure
         XmlStructNode rootStructNode = null;
         Node rootNode = null;

         // get the document node for XSD document
         Document doc = parser.getDocument();
         Node docElem = doc.getDocumentElement();
         NodeList nodes = docElem.getChildNodes();

         // search for the root element node
         for (int ii = 0; (ii < nodes.getLength() ) && ( null == rootStructNode) ;
ii++)
```

```
      {
         if (nodes.item(ii).getNodeType() == Node.ELEMENT_NODE )
         {
            NamedNodeMap attributes = nodes.item(ii).getAttributes();
            rootStructNode = new XmlStructNode(
attributes.getNamedItem("name").getNodeValue() );
            rootStructNode.setPath( rootStructNode.getNodeName() );
            rootStructNode.setNodeDepth(1);
            rootNode = nodes.item(ii);
         }
      }

      // now recursively build the structure from the root node onwards
      if ( !buildFromNode(rootStructNode, rootNode) )
      {
         if (DEBUG > 0) System.err.println("ERROR:\tbuildStructure()\n\tFailed
to build structure from root node!");
         return null;
      }

      return rootStructNode;

   }  // buildStructure()


   /**
    * A recursive method to build the structure from a particular node in the
    * XSD document.
    *
    * @param xsn    The current XmlStructNode in the structure that we are
building.
    * @param n      The current Xml node in the XSD document from which we are
building the
    *               structure.
    *
    * @return A boolean indicating the success (true) or failure (false) of the
    *         operation
    */
   private boolean buildFromNode(XmlStructNode xsn, Node n)
   {
      // get the list of child nodes
      NodeList nodes = n.getChildNodes();

      // for each child node in the XSD document
      for (int ii = 0; ii < nodes.getLength(); ii++)
      {
         if ( nodes.item(ii).getNodeName() == "xsd:element" )
         {
            // add a new entry in the structure for this element
            NamedNodeMap attributes = nodes.item(ii).getAttributes();
            XmlStructNode xsnSub = new XmlStructNode(
attributes.getNamedItem("name").getNodeValue() );
            xsn.addSubNode(xsnSub);
            xsnSub.setParentNode(xsn);
            xsnSub.setPath( xsn.getPath() + "/" + xsnSub.getNodeName() );
            xsnSub.setNodeDepth( xsn.getNodeDepth() + 1 );

            // get this node in the xsd document
            Node nSub = nodes.item(ii);
```

```
            // make the recursive call for the new sub node in the structure,
and the sub node in the XSD document
            if ( !buildFromNode(xsnSub, nSub) )
            {
               if (DEBUG > 0)
System.err.println("ERROR:\buildFromNode()\n\tFailed to build structure from
node!");
               return false;
            }
         }
         else
         {
            if ( !buildFromNode(xsn, nodes.item(ii) ) )
            {
               if (DEBUG > 0)
System.err.println("ERROR:\buildFromNode()\n\tFailed to build structure from
node!");
               return false;
            }
         }
      }

      return true;

   }  // buildFromNode()


   public boolean buildLinks()
   {
      return buildLinks(fromStructRoot, toStructRoot);
   }


   /**
    * This method attempts to recursively find the link between each component
of the input
    * structure and any of the components of the output structure.
    *
    * @param fromStructRoot
    *               The XmlStructNode that represents a the root the input
structure.
    * @param toStructNode
    *               The XmlStructNode that represents a component of the output
structure.
    *
    * @return A boolean that indicates the success (true) or failure (false) of
the
    *         operation.
    *
    */
   private boolean buildLinks(XmlStructNode fromStructRoot, XmlStructNode
toStructNode)
   {
      // search for the link for this component of the 'to' structure
      if ( !findLinks(fromStructRoot, toStructNode) )
      {
         System.err.println("ERROR:\buildLinks()\n\tFailed to find link for
node!");
         return false;
      }
```

```
      // now call this method recursively build the links for any subnodes
      for (int ii = 0; ii < toStructNode.getNumSubnodes(); ii++)
      {
         if ( !buildLinks(fromStructRoot, toStructNode.getSubNode(ii) ) )
         {
            return false;
         }
      }

      return true;

   }  // buildLinks()


   /**
    * This method attempts to recursively find the link between a single
component of the input
    * structure and any of the components of the output structure.
    *
    * @param fromStructNode
    *               The XmlStructNode that represents a component of the input
structure.
    * @param toStructNode
    *               The XmlStructNode that represents a component of the output
structure.
    *
    * @return A boolean that indicates the success (true) or failure (false) of
the
    *         operation.
    */
   private boolean findLinks(XmlStructNode fromStructNode, XmlStructNode
toStructNode)
   {
      if (DEBUG > 0)
      {
         System.out.println("Search for link for " +
toStructNode.getNodeName() + " at element " + fromStructNode.getNodeName() );
      }

      // do we have a match between these nodes?
      if ( fromStructNode.getNodeName().equals( toStructNode.getNodeName() ) )
      {
         // create a new link object, and add the 'to' and 'from' nodes
         XmlStructLink link = new XmlStructLink(toStructNode);
         if ( !link.addFromNode(fromStructNode) )
         {
            System.err.println("ERROR:\findLinks()\n\tFailed to add 'from'
link!");
            return false;
         }
         fromStructNode.setLink(link);
         toStructNode.setLink(link);

         // run back to the root of each struct tree, and set the descendant
links flags
         XmlStructNode n = toStructNode.getParentNode();
         while ( null != n )
         {
            n.setDescendantLinks(true);
```

```
            n = n.getParentNode();
         }
         n = fromStructNode.getParentNode();
         while ( null != n )
         {
            n.setDescendantLinks(true);
            n = n.getParentNode();
         }

         // no more recursion
         return true;
      }

      // no match, so recursively search all of the subnodes, assuming that
there are any
      for (int ii = 0; ( (ii < fromStructNode.getNumSubnodes() ) && (null ==
toStructNode.getLink() ) ); ii++)
      {
         if ( !findLinks(fromStructNode.getSubNode(ii), toStructNode) )
         {
            return false;
         }
      }

      return true;

   }  // findLinks()


   /**
    * This method builds the XSLT that will be used to convert documents
    * between the two XML formats.
    *
    * @param toStructRoot
    *
    * @return A boolean indicating the success (true) or failure (false) of the
    *         operation.
    *
    */
   private boolean buildXslt(XmlStructNode toStructRoot, XmlStructNode
fromStructRoot)
   {
      // write the start of the xslt document
      xsltWrite("<?xml version=\"1.0\"?>", 0);
      xsltWrite("<xsl:stylesheet version=\"1.0\"
xmlns:xsl=\"http://www.w3.org/1999/XSL/Transform\">\n", 0);
      xsltWrite("<xsl:output method=\"xml\" />\n", 1 );

      if ( !buildXsltTemplate(toStructRoot, fromStructRoot) )
      {
         System.err.println("ERROR:\tbuildXslt()\n\tFailed to build the XSLT
template!");
         return false;
      }

      if ( !writeXsltComponents() )
      {
         System.err.println("ERROR:\tbuildXslt()\n\tFailed to write the XSLT
components from the database!");
         return false;
```

```
        }

        // write the end of the xslt document
        xsltWrite("</xsl:stylesheet>", 0);

        return true;
    }



    private boolean writeXsltComponents()
    {
        boolean retVal = true;

        // database interaction object declarations
        Driver d = null;
        String url;
        Connection dbConn = null;
        String query = "";
        Statement stmt;
        ResultSet rs = null;

        // check that we have components to write
        if (xsltComponents.size() > 0)
        {
            try
            {
                String dbHost = "localhost";
                String dbName = "TM";
                String dbUser = "idact";
                String dbPass = "JX?^@Jh7CM";
                d = (Driver)Class.forName("com.mysql.jdbc.Driver").newInstance();
                dbConn = DriverManager.getConnection( ("jdbc:mysql://" + dbHost +
"/" + dbName +
                                                      "?user=" + dbUser + "&password="
+ dbPass) );
                //d =
(Driver)Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
                //url = "jdbc:odbc:" + "IDACT_DTM";
                //dbConn = DriverManager.getConnection(url, "", "");

                Iterator it = xsltComponents.iterator();

                while (it.hasNext())
                {
                    String componentName = (String)it.next();

                    //  build the query
                    query = "SELECT * FROM TM.XSLT_COMPONENTS WHERE NAME='" +
componentName + "'";

                    // execute
                    stmt = dbConn.createStatement();
                    rs = stmt.executeQuery(query);

                    // write this XSLT component, if it exists
                    if (rs.next())
                    {
                        xsltWrite(rs.getString("XSLT") + "\n", 0);
                    }
```

```
                else
                {
                    throw new Exception("XSLT component named '" +
componentName + "' not found in database.");
                }

            }
          }
          catch ( java.lang.InstantiationException ie)
          {
             System.err.println("ERROR:\twriteXsltComponents()\n\tFailed to
instantiate DB connection class!");
             retVal = false;
          }
          catch ( java.lang.IllegalAccessException iae)
          {
             System.err.println("ERROR:\twriteXsltComponents()\n\tIllegal
Access Exception!");
             retVal = false;
          }
          catch ( java.lang.ClassNotFoundException cnfe )
          {
             System.err.println("ERROR:\twriteXsltComponents()\n\tCannot find
database driver class!");
             retVal = false;
          }
          catch (java.sql.SQLException se)
          {
             System.err.println("ERROR:\twriteXsltComponents()\n\tSQL
Exception!");
             retVal = false;
          }
          catch (Exception e)
          {
             System.err.println("ERROR:\twriteXsltComponents()\n\t" +
e.getMessage());
             retVal = false;
          }
          finally
          {
             try
             {
                // close the DB connection if necessary
                if (dbConn != null && !dbConn.isClosed())
                {
                   dbConn.close();
                }

                // debug info
                if (DEBUG > 0)
                {
                   System.out.println("DEBUG\twriteXsltComponents");
                   System.out.println("\tquery='" + query + "'");
                   if (rs != null)
                   {
                      System.out.println( rs.getWarnings() );
                   }
                }
             }
             catch (java.sql.SQLException se)
```

```
            {
               System.err.println("ERROR:\twriteXsltComponents()\n\tSQL
Exception during cleanup!");
               retVal = false;
            }
         }
      }

      return retVal;
   }


   private boolean writeXsltTemplate(XmlStructFormatDescTemplate xmsdt,
XmlStructNode toStructNode, int indent)
   {
      // sanity check to ensure this is really a template type
      if ( XmlStructFormatDesc.TYPE_TEMPLATE != xmsdt.getType() )
      {
         System.out.println("ERROR\twriteXsltTemplate()\n\tNot a template
type!");
         return false;
      }

      if (xmsdt.getName().equals("CONCAT"))
      {
         // list each of the params in order
         for (int ii = 0; ii < xmsdt.getNumParams(); ii++)
         {
            switch (xmsdt.getParam(ii).getType())
            {
               case XmlStructFormatDesc.TYPE_ELEMENT:
                  // This is an element parameter, so get the path to the
element and use the value directly
                  String relativePath = buildRelativePath(
toStructNode.getLink().getFromNode(
((XmlStructFormatDescElement)xmsdt.getParam(ii)).getElementId()  ),

toStructNode.getParentNode().getLink().getFromNode(0) );
                  xsltWrite("<xsl:value-of select=\"" + relativePath + "\"
/>", indent );
                  break;
               case XmlStructFormatDesc.TYPE_STRING:
                  // This is a literal string, so enclose it in <xsl:text>
tags
                  xsltWrite("<xsl:text>" + (
(XmlStructFormatDescString)xmsdt.getParam(ii) ).getString() + "</xsl:text>",
indent);
                  break;
               case XmlStructFormatDesc.TYPE_TEMPLATE:
                  // we have another template, so make the recursive call
                  if (!writeXsltTemplate(
(XmlStructFormatDescTemplate)xmsdt.getParam(ii), toStructNode, indent + 1))
                  {
                     return false;
                  }
                  break;
               default:
                  System.out.println(indent +
"ERROR\twriteXsltTemplate()\n\tInvalid parameter type!");
                  return false;
```

```
                    //break;
                }
            }
        }
        else
        {
            // we are really going to call a template, and send params to it
            xsltWrite("<xsl:call-template name=\"" + xmsdt.getName() + "\" >",
indent);

            // list each of the params in order
            for (int ii = 0; ii < xmsdt.getNumParams(); ii++)
            {
                xsltWrite("<xsl:with-param name=\"p" + ii + "\">", indent + 1);
                switch (xmsdt.getParam(ii).getType())
                {
                    case XmlStructFormatDesc.TYPE_ELEMENT:
                        // This is an element parameter, so get the path to the
element and use the value directly
                        String relativePath = buildRelativePath(
toStructNode.getLink().getFromNode(
((XmlStructFormatDescElement)xmsdt.getParam(ii)).getElementId()  ),

toStructNode.getParentNode().getLink().getFromNode(0) );
                        xsltWrite("<xsl:value-of select=\"" + relativePath + "\"
/>", indent + 2 );
                        break;
                    case XmlStructFormatDesc.TYPE_STRING:
                        // This is a literal string, so enclose it in <xsl:text>
tags
                        xsltWrite("<xsl:text>" + (
(XmlStructFormatDescString)xmsdt.getParam(ii) ).getString() + "</xsl:text>",
indent + 2);
                        break;
                    case XmlStructFormatDesc.TYPE_TEMPLATE:
                        // we have another template, so make the recursive call
                        if (!writeXsltTemplate(
(XmlStructFormatDescTemplate)xmsdt.getParam(ii), toStructNode, indent + 1))
                        {
                            return false;
                        }
                        break;
                    default:
                        System.out.println(indent +
"ERROR\twriteXsltTemplate()\n\tInvalid parameter type!");
                        return false;
                        //break;
                }
                xsltWrite("</xsl:with-param>", indent + 1);
            }
            xsltWrite("</xsl:call-template>", indent);

            // add this template to the list to be added to the XSLT output
            xsltComponents.add( xmsdt.getName() );
        }
        return true;
    }

    /**
     * This method builds the XSLT template for a given node in the 'to'
```

```
     * structure.
     *
     * @param toStructNode
     *               A node in the 'to' structure
     *
     * @return A boolean indicating the success (true) or failure (false) of the
     *         operation.
     */
    private boolean buildXsltTemplate(XmlStructNode toStructNode, XmlStructNode
fromStructRoot)
    {

       // deal with the root node stuff, if this is the root node
       if ( null == toStructNode.getParentNode() )
       {
          // write the root node stuff
          xsltWrite("<xsl:template match=\"/\">", 1 );
          xsltWrite("<xsl:apply-templates select=\"" +
toStructNode.getLink().getFromNode(0).getPath() + "\" />", 2 );
          xsltWrite("</xsl:template>\n", 1);

          // set the path this node
          //String pathToNode =
toStructNode.getLink().getFromNode(0).getPath();
       }

       // if there are no subnodes, then create this element using the 'from'
values.
       if ( !toStructNode.getDescendantLinks() )
       {
          // start the XSLT template, using the call convention
          xsltWrite("<xsl:template name=\"" +
toStructNode.getLink().getFromNode(0).getNodeName() + "_to_" +
                   toStructNode.getNodeName() + "\" >", 1);
          xsltWrite("<xsl:element name=\"" + toStructNode.getNodeName() + "\"
>", 2);

          if (!writeXsltTemplate(toStructNode.getLink().getFormatNode(),
toStructNode, 3))
          {
             return false;
          }

          // complete this element
          xsltWrite("</xsl:element>", 2);
       }
       else
       {
          // start the XSLT template, using an apply convention
          xsltWrite("<xsl:template match=\"" +
toStructNode.getLink().getFromNode(0).getNodeName() + "\" >", 1);

          // start this element
          xsltWrite("<xsl:element name=\"" + toStructNode.getNodeName() + "\"
>", 2);

          // apply the template for each subnode
          for (int ii = 0; ii < toStructNode.getNumSubnodes(); ii++)
          {
             // attempt to build the relative path to this subnode
```

```
            String relativePath = buildRelativePath(
toStructNode.getSubNode(ii).getLink().getFromNode(0),
toStructNode.getLink().getFromNode(0) );
            if ( null == relativePath)
            {
               System.err.println("ERROR:\buildXsltTemplate()\n\tFailed to
build the relative path!");
               return false;
            }

            // now write the template call for this subnode, including the
sorts
            if ( toStructNode.getSubNode(ii).getLink().getSort().getNumSorts()
> 0 )
            {
               // include the sorts in the template call
               xsltWrite("<xsl:apply-templates select=\"" + relativePath +
"\">", 3);
               for (int jj = 0; jj <
toStructNode.getSubNode(ii).getLink().getSort().getNumSorts(); jj++)
               {
                  relativePath = buildRelativePath(
toStructNode.getSubNode(ii).getLink().getSort().getSortNode(jj),
toStructNode.getSubNode(ii).getLink().getFromNode(0) );
                  String sortOrder =
toStructNode.getSubNode(ii).getLink().getSort().getSortOrder(jj);
                  String tempXslt = "<xsl:sort select=\"" + relativePath +
"\" ";
                  if (null != sortOrder)
                  {
                     tempXslt += "order=\"" + sortOrder + "\" ";
                  }
                  xsltWrite(tempXslt + " />", 4);
               }
               xsltWrite("</xsl:apply-templates>", 3);
            }
            else
            {
               // no sorts, so just call or apply the template
               if ( !toStructNode.getSubNode(ii).getDescendantLinks() )
               {
                  xsltWrite("<xsl:call-template name=\"" +
toStructNode.getSubNode(ii).getLink().getFromNode(0).getNodeName() +
                           "_to_" + toStructNode.getSubNode(ii).getNodeName()
+ "\" />", 3);
               }
               else
               {
                  xsltWrite("<xsl:apply-templates select=\"" + relativePath +
"\" />", 3);
               }
            }
         }

         // end this element
         xsltWrite("</xsl:element>", 2);

      }
      // end the template for this node
      xsltWrite("</xsl:template>\n", 1);
```

```
      // write all of the templates for the subelements
      for (int ii = 0; ii < toStructNode.getNumSubnodes(); ii++)
      {
         if ( !buildXsltTemplate(toStructNode.getSubNode(ii), fromStructRoot )
)
         {
            return false;
         }
      }

      return true;
   }


   /**
    * This method builds the relative XPath between two nodes in the 'from'
    * structure.
    *
    * @param n1     A XMLStructNode that the node that we're building the
relative XPath to.
    * @param n2     A XMLStructNode that the node that we're building the
relative XPath from.
    *
    * @return A String that contains the relative XPath, or an empty string in
the
    *         even of an error.
    *
    */
   private String buildRelativePath(XmlStructNode n1, XmlStructNode n2)
   {
      String relativePath = new String("");

      // get the absolute path to this node, and to the parent
      String fullPathToNode1 = n1.getPath();
      String fullPathToNode2 = n2.getPath();

      if (DEBUG > 0)
      {
         System.out.println("Attempting to find relative path between " +
n1.getNodeName() + " and " + n2.getNodeName());
         System.out.println(fullPathToNode1);
         System.out.println(fullPathToNode2);
      }

      // do we have the same node?
      if ( fullPathToNode1.equals(fullPathToNode2) )
      {
         relativePath = ".";
      }
      else
      {
         // tokenise the path to this node and the parent node
         String pathTokensNode1[] = fullPathToNode1.split("/");
         String pathTokensNode2[] = fullPathToNode2.split("/");

         // now find the part of the absolute paths that match
         int pathMatchDepth = 0;
         boolean pathMismatchFound = false;
```

```
          while ( (!pathMismatchFound) && (pathMatchDepth <
pathTokensNode1.length) && (pathMatchDepth < pathTokensNode2.length) )
          {
             if (
!pathTokensNode1[pathMatchDepth].equals(pathTokensNode2[pathMatchDepth]) )
             {
                pathMismatchFound = true;
             }
             else
             {
                pathMatchDepth++;
             }
          }

          // if no mismatch found, then one node is a direct descendant of the
other
          if ( !pathMismatchFound )
          {
             if (pathTokensNode1.length > pathTokensNode2.length)
             {
                if (DEBUG > 0) System.out.println("Code path 1");
                // node 1 a direct descendant of node 2
                for (int ii = pathMatchDepth; ii < (pathTokensNode1.length -
1); ii++)
                {
                   relativePath += pathTokensNode1[ii] + "/";
                }
                relativePath += pathTokensNode1[pathTokensNode1.length - 1];
             }
             else
             {
                if (DEBUG > 0) System.out.println("Code path 2");
                // node 2 a direct descendant of node 1
                for (int ii = pathTokensNode2.length - 1; ii > pathMatchDepth;
ii--)
                {
                   relativePath += "../";
                }
             }
          }
          else
          {
             if (DEBUG > 0)
             {
                System.out.println("Code path 3");
                System.out.println("pathTokensNode2.length - 1 = " +
(pathTokensNode2.length - 1) + ", pathMatchDepth = " + pathMatchDepth);
             }
             // we have to go up in the tree from node 2
             for (int ii = pathTokensNode2.length; ii > pathMatchDepth; ii--)
             {
                relativePath += "../";
             }

             // now back down to node 1
             for (int ii = pathMatchDepth; ii < (pathTokensNode1.length - 1);
ii++)
             {
                relativePath += pathTokensNode1[ii] + "/";
             }
```

```
            relativePath += pathTokensNode1[pathTokensNode1.length - 1];
         }
      }

      if (DEBUG > 0) System.out.println("Relative path found was '" +
relativePath + "'\n");
      return relativePath;

   }  // buildRelativePath()


   /**
    * This method appends the latest chunk of XSLT to a string, which will
    * later be used to write the entire XSLT file.
    *
    * @param s      A String containing the latest line (chunk) of XSLT.
    * @param level  The indentation level at which to write this chunk of XSLT.
    *
    */
   private void xsltWrite(String s, int level)
   {
      String indent = "";
      for (int ii = 0; ii < level; ii++)
      {
         indent += "\t";
      }
      xsltString += indent + s + "\n";

   }  // xsltWrite()


   /**
    * This method attempts to open a file, and if successful it write the
contents
    * of the xsltString object to the file.  Essentially, we're writing the
    * XSLT file.
    *
    * @param xsltFilename
    *               The name of the XSLT file to be written.
    *
    * @return A boolean indicating the success (true) or failure (false) of the
operation.
    */
   private boolean writeXsltFile(String xsltFilename)
   {
      try
      {
         BufferedWriter out = new BufferedWriter( new FileWriter
(xsltFilename) );
         out.write( xsltString );
         out.close();
      }
      catch (IOException ioe)
      {
         System.err.println("ERROR:\twriteXsltFile()\n\tIOException!");
         return false;
      }

      return true;
   }
```

```
   /**
    * A recursive method that displays the XSD structure.
    *
    * @param xsn    The current node in the structure to be displayed.
    * @param level  The hierarchical level of the current node (controls the
level of
    *               indentation used in the output).
    */
   private void printStruct(XmlStructNode xsn)
   {
      if (xsn != null)
      {
         // print the indent
         for (int ii = 1; ii < xsn.getNodeDepth(); ii++)
         {
            System.out.print("  ");
         }

         // print the name of, and the path to, this node
         System.out.print( xsn.getNodeName() );
         System.out.println( " (" + xsn.getNodeDepth() + " - " + xsn.getPath()
+ ")" );

         // print the subnodes, at the next indent level
         for (int ii = 0; ii < xsn.getNumSubnodes(); ii++)
         {
            printStruct( xsn.getSubNode(ii) );
         }
      }

   }  // printStruct()


   /**
    * A recursive method that displays the links found between the 'from' and
    * 'to' structures.
    *
    * @param xsn    The current node in the 'to' structure, whose link we will
display.
    *
    */
   private void printLinks(XmlStructNode xsn)
   {
      if (xsn != null)
      {
         // print the indent
         for (int ii = 1; ii < xsn.getNodeDepth(); ii++)
         {
            System.out.print("  ");
         }

         // print the name of this node
         System.out.print( xsn.getNodeName() );

         // print the link info
         if ( null != xsn.getLink() )
         {
            System.out.print(" links to " +
```

```
xsn.getLink().getFromNode(0).getNodeName() );
             for (int ii = 1; ii < xsn.getLink().getNumFromNodes(); ii++)
             {
                System.out.print( ", " +
xsn.getLink().getFromNode(ii).getNodeName() );
             }

             if (xsn.getLink().getSort().getNumSorts() > 0)
             {
                System.out.print(", sorted by " +
xsn.getLink().getSort().getSortNode(0).getNodeName() );
                for (int ii = 1; ii < xsn.getLink().getSort().getNumSorts();
ii++)
                {
                   System.out.print( ", " +
xsn.getLink().getSort().getSortNode(ii).getNodeName() );
                }
             }
          }
          else
          {
             System.out.print(" - no link found! ");
          }

          System.out.println();

          // print the subnodes, at the next indent level
          for (int ii = 0; ii < xsn.getNumSubnodes(); ii++)
          {
             printLinks(xsn.getSubNode(ii) );
          }
       }

   }  // printLinks()



   /**
    * This method allows the user to manually build links.
    *
    * @param fromStructRoot
    *               The XmlStructNode that represents a the root the input
structure.
    * @param toStructNode
    *               The XmlStructNode that represents a component of the output
structure.
    *
    * @return A boolean that indicates the success (true) or failure (false) of
the
    *         operation.
    */
   private boolean buildLinksManual(XmlStructNode fromStructRoot, XmlStructNode
toStructRoot)
   {
      try
      {
         boolean done = false;
         BufferedReader keyboard = new BufferedReader( new InputStreamReader(
System.in ) );
         String line, fromLine, toLine;
```

```
          do
          {
             // ask if we're adding links
             System.out.print("\nManually add link (y/n): ");
             line = keyboard.readLine();
             if ( line.startsWith("y") )
             {
                // display the 'from', 'to', and current-link structures
                System.out.println("'From' stucture\n--------------");
                printStruct(fromStructRoot);
                System.out.println("\n'To' stucture\n------------");
                printStruct(toStructRoot);
                System.out.println("\nOutput Link Structure\n------------------
---");
                printLinks(toStructRoot);

                // ask the user for the 'from' field
                System.out.print("Enter 'from' field: ");
                fromLine = keyboard.readLine();
                String[] fromFields = fromLine.split(",");

                // ask the user for the 'to' field
                System.out.print("Enter 'to' field: ");
                toLine = keyboard.readLine();
                String[] toFields = toLine.split(",");

                // find the nodes to link together
                XmlStructNode fromNode = fromStructRoot;
                for (int ii = 0; ( (fromLine.length() > 0) && (ii <
fromFields.length) && (fromNode != null) ); ii++)
                {
                   try
                   {
                      int jj = Integer.parseInt( fromFields[ii] );
                      fromNode = fromNode.getSubNode(jj);

                   }
                   catch (NumberFormatException nfe)
                   {
                      System.out.println("Invalid path to 'from' node.");
                      fromNode = null;
                   }
                }
                XmlStructNode toNode = toStructRoot;
                for (int ii = 0; ( (toLine.length() > 0) && (ii <
toFields.length) && (toNode != null) ); ii++)
                {
                   try
                   {
                      int jj = Integer.parseInt( toFields[ii] );
                      toNode = toNode.getSubNode(jj);

                   }
                   catch (NumberFormatException nfe)
                   {
                      System.out.println("Invalid path to 'to' node.");
                      toNode = null;
                   }
                }
```

```
                // if both nodes were valid
                if ( ( fromNode != null ) && ( toNode != null ) )
                {
                   // create a link object, if necessary
                   XmlStructLink link = toNode.getLink();
                   if (null == link)
                   {
                      link = new XmlStructLink(toNode);
                      toNode.setLink(link);
                   }

                   link.addFromNode(fromNode);
                }
                else
                {
                   System.err.println("ERROR:\buildLinksManual()\n\tInvalid
node." );
                   return false;
                }
             }
             else
             {
                done = true;
             }
          } while ( !done );

       }
       catch (IOException ioe)
       {
          System.err.println("ERROR:\buildLinksManual()\n\tIOException\n\t" +
ioe.getMessage() );
          return false;
       }

       return true;

   }  // buildLinksManual();


   /**
    * This method allows the user to manually build sorts.
    *
    * @param fromStructRoot
    *               The XmlStructNode that represents a the root the input
structure.
    * @param toStructNode
    *               The XmlStructNode that represents a component of the output
structure.
    *
    * @return A boolean that indicates the success (true) or failure (false) of
the
    *         operation.
    */
   private boolean buildSortsManual(XmlStructNode fromStructRoot, XmlStructNode
toStructRoot)
   {

       try
       {
```

```
          boolean done = false;
          BufferedReader keyboard = new BufferedReader( new InputStreamReader(
System.in ) );
          String line, fromLine, toLine;

          do
          {
              // ask if we're adding links
              System.out.print("\nManually add sort (y/n): ");
              line = keyboard.readLine();
              if ( line.startsWith("y") )
              {
                  // ask the user for the 'to' field
                  System.out.println("\n'To' stucture\n------------");
                  printStruct(toStructRoot);
                  System.out.print("Enter 'to' field: ");
                  toLine = keyboard.readLine();
                  String[] toFields = toLine.split(",");

                  // is this a valid 'to' field?
                  XmlStructNode toNode = toStructRoot;
                  for (int ii = 0; ( (toLine.length() > 0) && (ii <
toFields.length) && (toNode != null) ); ii++)
                  {
                      try
                      {
                          int jj = Integer.parseInt( toFields[ii] );
                          toNode = toNode.getSubNode(jj);
                      }
                      catch (NumberFormatException nfe)
                      {
                          System.out.println("Invalid path to 'to' node.");
                          toNode = null;
                      }
                  }
                  if (null != toNode)
                  {
                      // is there a link for this 'to' field
                      XmlStructLink link = toNode.getLink();
                      if (null != link)
                      {
                          // ask the user to enter the linked 'from' field to sort
                          for (int ii = 0; ii < link.getNumFromNodes(); ii++)
                          {
                              System.out.println(ii + ": " +
link.getFromNode(ii).getPath() );
                          }
                          System.out.print("\nEnter the node to sort: ");
                          line = keyboard.readLine();

                          // convert the value to an int
                          XmlStructNode fromNode = null;
                          try
                          {
                              int fromNum = Integer.parseInt(line);
                              fromNode = link.getFromNode(fromNum);
                          }
                          catch (NumberFormatException nfe)
                          {
                              System.out.println("Invalid 'from' node.");
```

```
                        fromNode = null;
                    }

                    // if this is a valid 'from' node
                    if ( null != fromNode )
                    {
                        // ask the user for the sort field
                        printStruct(fromStructRoot);
                        System.out.print("\nEnter the node to sort on: ");
                        line = keyboard.readLine();
                        String[] sortFields = line.split(",");

                        // is this a valid 'sort' field?
                        XmlStructNode sortNode = fromStructRoot;
                        for (int ii = 0; ( (line.length() > 0) && (ii <
sortFields.length ) && (sortNode != null) ); ii++)
                        {
                            try
                            {
                                int jj = Integer.parseInt( sortFields[ii] );
                                sortNode = sortNode.getSubNode(jj);
                            }
                            catch (NumberFormatException nfe)
                            {
                                System.out.println("Invalid path to 'sort'
node.");
                                sortNode = null;
                            }
                        }

                        // do we have a valid 'sort' node?
                        if (sortNode != null)
                        {
                            // ask the user if we're sorting in ascending or
descending order
                            System.out.print("Enter sort order (asc/desc):
");
                            line = keyboard.readLine();

                            String sortOrder = null;
                            if ( line.startsWith("asc") )
                            {
                                sortOrder = new
String(XmlStructSort.ORDER_ASCEND);
                            }
                            else if ( line.startsWith("desc") )
                            {
                                sortOrder = new
String(XmlStructSort.ORDER_DESCEND);
                            }

                            // if we have a valid sort order
                            if (null != sortOrder)
                            {
                                // add the sort
                                if (!link.getSort().addSort(sortNode,
sortOrder))
                                {
                                    // adding the sort failed
                                    return false;
```

```
                                }
                            }
                        }
                    }
                }
            }
            else
            {
                done = true;
            }
        } while ( !done );

    }
    catch (IOException ioe)
    {
        System.err.println("ERROR:\buildLinksManual()\n\tIOException\n\t" +
ioe.getMessage() );
        return false;
    }

    return true;

}  // buildSortsManual()

}  // class XsltGenerator
```

```
/*
 * Copyright 2004 Brian Hay (IDACT)
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */


// imports for input
import java.io.BufferedReader;
import java.io.InputStreamReader;

// imports for the Vector class
import java.util.Vector;


/**
 * A Command Line Interface for the prototype XSLT Generator.
 *
 * @author Brian Hay
 *
 * @version $Id: XsltGeneratorCli.java,v 1.2 2005/01/03 08:09:36 bhay Exp $
 *
 */
public class XsltGeneratorCli
{
   /**
    * private class members
    */
   private XsltGenerator xgen;          // XsltGenerator object
   private BufferedReader keyboard;  // input from keyboard
   private final int DEBUG = 0;      // debug level - greater than zero causes
debug messages
   private boolean readyToWriteXslt;// indicates if we have met all of the
conditions necessary
                                    // to write the XSLT output.

   /**
    * private constants
    */
   //private final char FIELD_DELIM = ',';    // delimiter for the 'to' and
'from' field levels
   //private final String NO_MORE_FIELDS = "!";   // indicator that there are
no more fields to enter


   /**
    * This method provides the top level of control for the CLI.  Essentially
it
```

```
     * calls each step in the process in order, until the XSLT is generated or
     * an exception occurs.
     */
    public void goMenu()
    {
        boolean done = false;      // boolean to control menu loop
        readyToWriteXslt = false;

        try
        {
            // get the input and output format filename, and the XSLT filename,
and use
            // them to create a new XsltGenerator object
            createXsltGen();

            do
            {
                // display the menu and get the selection
                readyToWriteXslt = showMenu();
                int choice = getChoice();

                switch (choice)
                {
                    case 0:
                        // exit the menu
                        done = true;
                        break;
                    case 1:
                        // print the input and output structures
                        showInputStruct();
                        showOutputStruct();
                        showLinks();
                        break;
                    case 2:
                        // allow the user to add a new link
                        addLink();
                        xgen.showLinks();
                        break;
                    case 3:
                        // allow the user to delete a link
                        deleteLink();
                        xgen.showLinks();
                        break;
                    case 9:
                        // write XSLT
                        if (readyToWriteXslt)
                        {
                            writeXslt();
                            break;
                        }
                    default:
                        System.out.println("Invalid input!");
                }

            } while (!done);
        }
        catch (Exception e)
        {
            System.out.println(e.getMessage());
            return;
```

```
      }
   }

   /**
    * This method attempts to write the XSLT document to a file.
    *
    * @exception Exception
    */
   private void writeXslt() throws Exception
   {
      if (!xgen.writeXsltDoc())
      {
         if (DEBUG > 0) System.err.println("DEBUG:\twriteXslt()\n\tFailed to
write XSLT document.");
         throw new Exception("Failed to write XSLT document");
      }
   }

   /**
    * This method allows the user to enter the input ('from') and output
('from')
    * fields to be linked, and then attempts to create that link.
    *
    * @exception Exception
    */
   private void addLink() throws Exception
   {
      Vector inputFields = new Vector();

      // get as many 'from' fields as necessary, and add them to the vector
      do
      {
         inputFields.add( getNodePath(XsltGenerator.FROM_STRUCT) );
      } while ( getYesNoInput("", "Add another input field (y/n): ") );

      // get the 'to' field
      Vector outputField = getNodePath(XsltGenerator.TO_STRUCT);

      // attempt to add the link
      xgen.addNewLink(outputField, inputFields);

      // define the TCs for the link
      xgen.addLinkDef(outputField);
   }


   //private void addTransComponent() throws Excpetion
   //{
      // get the 'to' field
      //Vector outputField = getNodePath(XsltGenerator.TO_STRUCT);
   //}

   /**
    * This method allows the user to specify a link between 'to' and 'from'
fields
    * that will be deleted.
    *
    * @exception Exception
    */
   private void deleteLink() throws Exception
```

```
   {
      // get the 'to' field
      Vector outputField = getNodePath(XsltGenerator.TO_STRUCT);

      // are we deleting the entire link, or just part of it?
      if ( getYesNoInput("", "Delete entire link (y/n): ") )
      {
         xgen.deleteEntireLink(outputField);
      }
      else
      {
         // get the 'from' field, and delete that part of the link
         int fromNodeChoice = selectLinkedNode(outputField);
         if (0 != fromNodeChoice)
         {
            xgen.deleteLink(outputField, fromNodeChoice - 1);
         }
      }
   }


   /**
    * This method allows the user to select a 'from' node that is linked to a
    * given 'to' node.
    *
    * @param toNode An XmlStructNode that repesents the 'to' node.
    *
    * @return An XmlStructNode that represents the 'from' node selected, or
null if no
    *         node was selected.
    */
   private int selectLinkedNode(Vector outputField) throws Exception
   {
      if (null == outputField)
      {
         if (DEBUG > 0)
System.err.println("ERROR:\tselectLinkedNode()\n\toutputField parameter is
null.");
         throw new Exception("Undefined output field while attempting to
delete link");
      }

      Vector fromFieldNames = xgen.getLinkedNodeNames(outputField);

      if ( (null == fromFieldNames) || (fromFieldNames.size() < 1) )
      {
         System.out.println("No links found for 'to' node.");
         return 0;
      }

      for (int ii = 0; ii < fromFieldNames.size(); ii++)
      {
         System.out.println( (ii + 1) + " - " +
(String)fromFieldNames.elementAt(ii) );
      }
      System.out.println("\n0 - Don't delete a link.");

      return getIntInput("", "Enter choice in the range [0, " +
fromFieldNames.size() + "] : ", 0, fromFieldNames.size() );
   }
```

```
   /**
    * This method prompts the user for an 'y' or 'n' entry, accepts their
input,
    * and loops until they provide a valid response.
    *
    * @param indent A String that allows the prompt to be indented.
    * @param prompt The prompt to be displayed to the user prior to accepting
input.
    *
    * @return A boolean, with true indicating that 'y' was entered, and false
indicating
    *         that 'n' was entered.
    * @exception Exception
    */
   private boolean getYesNoInput(String indent, String prompt) throws Exception
   {
      try
      {
         do
         {
            System.out.print(indent + prompt);
            String s = keyboard.readLine().toLowerCase();
            if ( (null == s) || (s.length() != 1) )
            {
               System.out.println(indent + "Invalid entry");
            }
            else if (s.equals("y"))
            {
               return true;
            }
            else if (s.equals("n"))
            {
               return false;
            }
         } while (true);
      }
      catch (java.io.IOException ioe)
      {
         if (DEBUG > 0)
System.err.println("DEBUG:\tgetYesNoInput()\n\tIOException when reading y/n
response.\n\tPrompt'"+prompt+ "'\n\t"+ioe.getMessage());
         throw new Exception("Failed to read y/n response");
      }
   }


   /**
    * This method prompts the user for an integer entry, accepts their input,
    * and loops until they provide a valid response.  The integer will only be
    * accepted if it is in the range [minVal, maxVal].
    *
    * @param indent A String that allows the prompt to be indented.
    * @param prompt The prompt to be displayed to the user prior to accepting
input.
    * @param minVal The minimum acceptable value that may be entered.
    * @param maxVal The maximum acceptable value that may be entered.
    *
    * @return The int value entered by the user.
    *
    * @exception Exception
```

```
    */
   private int getIntInput(String indent, String prompt, int minVal, int
maxVal) throws Exception
   {
      int choice = minVal - 1;
      boolean done;
      do
      {
         done = true;
         System.out.print(prompt);
         String s;
         try
         {
            s = keyboard.readLine();

            choice = Integer.parseInt(s);
            if ( (choice > maxVal) || (choice < minVal ) )
            {
               System.out.println(indent + "Invalid choice (out of range) -
please try again");
               done = false;
            }
         }
         catch (java.io.IOException ioe)
         {
            if (DEBUG > 0)
System.err.println("DEBUG:\tgetIntInput()\n\tIOException when reading
input.\n\t"+ioe.getMessage());
            throw new Exception("Failed to read int input");
         }
         catch (NumberFormatException nfe)
         {
            System.out.println(indent + "Invalid choice (not an integer) -
please try again");
            done = false;
         }
      } while (!done);

      return choice;
   }


   /**
    * This method allows the user to identify a node, either in the input or
    * output structure.
    *
    * @param ioStruct An int value that indicates whether the node will be in
the input
    *                 (XmlStructNode.FROM_STRUCT) or output
(XmlStructNode.TO_STRUCT) structure.
    *
    * @return A Vector containing the path to the chosen node.
    * @exception Exception
    */
   private Vector getNodePath(int ioStruct) throws Exception
   {
      if ( (ioStruct != XsltGenerator.FROM_STRUCT) && (ioStruct !=
XsltGenerator.TO_STRUCT) )
      {
         if (DEBUG > 0) System.err.println("DEBUG:\tgetNodePath()\n\tParameter
```

```
ioStruct invalid (" + ioStruct + ")");
          throw new Exception("No such structure (" + ioStruct + ")");
      }

      String indent = "";
      Vector nodePath = new Vector();
      Vector nodeNames = xgen.getSubNodeNames(nodePath, ioStruct);
      boolean choiceMade = false;

      do
      {
          // allow the user to choose what to do
          for (int ii = 0; ii < nodeNames.size(); ii++)
          {
              System.out.println( indent + (ii+1) + " - " +
(String)nodeNames.elementAt(ii) );
          }
          System.out.println();
          System.out.println(indent + "0 - Move up a level");
          System.out.println();
          int choice = getNodeChoice(nodeNames.size(), indent);

          // act based on the user's choice
          if (choice > 0)
          {
              // attempt to move down a level in the hierarchy
              nodePath.add( new Integer(choice - 1) );
              Vector newSubNodeNames = xgen.getSubNodeNames(nodePath, ioStruct);
              if (null != newSubNodeNames)
              {
                  nodeNames = newSubNodeNames;
                  indent = indent + "  ";
              }
              else
              {
                  nodePath.removeElementAt( nodePath.size() - 1);
                  System.out.println(indent + "The node you selected has no
subnodes - please try again");
              }
          }
          else if (choice < 0)
          {
              // add the chosen node to the vector of 'from' fields
              nodePath.add( new Integer( (choice + 1) * -1));
              choiceMade = true;
          }
          else
          {
              // move up a level if possible
              if ( nodePath.size() > 0 )
              {
                  nodePath.removeElementAt( nodePath.size() - 1);
                  nodeNames = xgen.getSubNodeNames(nodePath, ioStruct);
                  indent = indent.substring(2);
              }
              else
              {
                  System.out.println("Already at the top level");
              }
          }
```

```
      } while (!choiceMade);

      return nodePath;
   }


   /**
    * This method allows the user to enter a node choice.  The range of valid
    * options is [ -maxVal, maxVal ].
    *
    * @param maxVal A positive value that is used to define the valid range of
entries to
    *               accept, which is [-maxValue, maxValue].
    *
    * @return An int containing the valid entry provided by the user.
    *
    * @exception Exception
    */
   private int getNodeChoice(int maxVal, String indent) throws Exception
   {
      boolean done;
      int choice = Integer.MIN_VALUE;
      if (null == indent)
      {
         indent = "";
      }

      return getIntInput(indent, "Enter choice in the range [" + (maxVal * -1)
+ ", " + maxVal + "] : ", (maxVal * -1), maxVal);
   }



   /**
    * This method displays the structure of the input format.
    */
   private void showLinks()
   {
      System.out.println();
      System.out.println("Links");
      System.out.println("-----");
      xgen.showLinks();
   }


   /**
    * This method displays the structure of the input format.
    */
   private void showInputStruct()
   {
      System.out.println();
      System.out.println("Input Structure");
      System.out.println("---------------");
      if ( !xgen.printFromStruct() )
      {
         System.out.println("No input structure defined!");
      }
   }
```

```
   /**
    * This method displays the structure of the output format.
    */
   private void showOutputStruct()
   {
      System.out.println();
      System.out.println("Output Structure");
      System.out.println("----------------");
      if ( !xgen.printToStruct() )
      {
         System.out.println("No output structure defined!");
      }
   }



   /**
    * This method attempts to read the user's choice from the keyboard.
    *
    * @return An int containing the integer value entered by the user, or -1 if
the
    *         user does not enter any input, and -2 if the user enters a value
that is
    *         not a number.
    * @exception Exception Exception is thrown if an error occurs while
attempting to read from the
    *                      keyboard.
    */
   private int getChoice() throws Exception
   {
      try
      {
         System.out.print("Enter your choice: ");
         String choice = keyboard.readLine();
         if ( (null == choice) || (choice.length() < 1))
         {
            return -1;
         }
         else
         {
            int c = Integer.parseInt(choice);
            return c;
         }
      }
      catch (java.io.IOException ioe)
      {
         if (DEBUG > 0) System.err.println("DEBUG:\tgetChoice()\n\tIOException
while attempting to read menu choice.\n\t"+ioe.getMessage());
         throw new Exception("Failed to read menu choice");
      }
      catch (java.lang.NumberFormatException nfe)
      {
         if (DEBUG > 0)
System.err.println("DEBUG:\tgetChoice()\n\tNumberFormatException while
attempting to read menu choice.\n\t"+nfe.getMessage());
         return -2;
      }
   }
```

```
    /**
     * This method displays the menu on the screen.  It also determine if we
     * have fullfilled the requirements to proceed to XSLT generation - if we
     * have it offers that as a menu option.
     *
     * @return A boolean that indicates if we are ready to continue to XSLT
     *         generation or not.
     */
    private boolean showMenu()
    {
       boolean retVal = false;

       System.out.println();
       System.out.println("1) Display the input and output structures");
       System.out.println("2) Add a new link");
       System.out.println("3) Delete a link");
       if (xgen.readyForXslt())
       {
          System.out.println("9) Generate XSLT");
          retVal = true;
       }
       System.out.println();
       System.out.println("0) Exit the program");
       System.out.println();

       return retVal;
    }


    /**
     * This method creates the input reader (keyboard), gets the filenames for
     * the input and output formats, and the XSLT output document, and creates
     * the XsltGenerator object using those values.
     */
    private void createXsltGen() throws Exception
    {
       String inFormFile, outFormFile, xsltFile;
       keyboard = new BufferedReader( new InputStreamReader( System.in ) );

       // get the filenames from the user
       //try
       //{
          inFormFile = "/idact/formats/fromG.xsd";
          outFormFile = "/idact/formats/toG.xsd";
          xsltFile = "/idact/transformations/fromG_to_toG.xsd";
          //System.out.print("Enter input format filename:  ");
          //inFormFile = keyboard.readLine();
          //System.out.print("Enter output format filename: ");
          //outFormFile = keyboard.readLine();
          //System.out.print("Enter XSLT output filename:   ");
          //xsltFile = keyboard.readLine();
       //}
       //catch (java.io.IOException ioe)
       //{
       // if (DEBUG > 0)
System.err.println("DEBUG:\tcreateXsltGen()\n\tIOException while attempting to
read filenames\n\t" + ioe.getMessage());
       // throw new Exception("Failed to read filenames");
       //}
```

```
        // create the xsltGenerator
        xgen = new XsltGenerator(inFormFile, outFormFile, xsltFile);

        xgen.buildLinks();
    }



    /**
     * main method, which is just used to get us going.
     *
     * @param args
     */
    public static void main(String[] args)
    {
        // create a CLI object and get it started
        XsltGeneratorCli xCli = new XsltGeneratorCli();
        xCli.goMenu();
    }

}
```